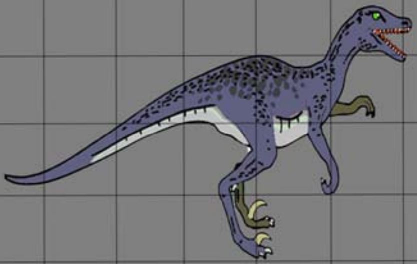
MATA AMRITANANDAMAYI MATH

श्रद्धावान् लभते ज्ञानम्

AMRITA VISHWA VIDYAPEETHAM UNIVERSITY

Established u/s 3 of UGC Act 1956

Computational Thinking For PROBLEM SOLVING



Amrita Vishwa Vidyapeetham LAB MANUAL Dept. of Computer Science & Engg. - Ettimadai

Table of Contents

1. TOOL DESCRIPTIONS : SPREAD SHEETS, SCRATCH AND RAPTOR

1.1. Spreadsheets

1.1.1. What are Spreadsheets and why Spreadsheets?

A spread sheet is an interactive computer application program for organization, analysis and storage of data in tabular form. Spreadsheets were developed as computerized versions of paper accounting worksheets. Spreadsheets have replaced paper-based systems throughout the business world. Although they were first developed for accounting or bookkeeping tasks, they are now used extensively in any context where tabular lists are built, sorted, and shared.

The program operates on data represented as cells of an array, organized in rows and columns. Each cell of the array may contain either numeric or text data, or the results of formulae that automatically calculate and display a value based on the contents of other cells. Spreadsheet allows users to adjust any stored value and observe the effects on calculated values thereby making spreadsheets a very useful tool for "what-if" analysis as a large amount of data of various cases can be quickly investigated without manual recalculation. Modern spreadsheet software can have multiple sheets which can interact with each other and can display data either as text, numerals, or in graphical form.

In addition to performing basic arithmetic and mathematical calculation, modern spreadsheets also provide built-in calculations for most frequently and commonly used financial and statistical operations. Calculations such as net present value or standard deviation can be applied to tabular data with pre-programmed calculations in a formula. Spreadsheet programs also provide conditional expressions, built-in options to convert between text and numbers, and options that operate on strings of text.

1.1.2. Flavors of Spreadsheet Software

Microsoft Excel is a proprietary spread sheet application developed by Microsoft for Microsoft Windows Operating Systems. Gnumeric is a free, cross-platform spreadsheet program that is part of the GNOME Free Software Desktop Project. OpenOffice Calc and the very closely related LibreOffice Calc are free and open-source spreadsheets. With the advent of advanced web technologies such as Ajax circa 2005, a new generation of online spreadsheets has emerged. Equipped with a rich Internet application user experience, the best web based online spreadsheets have many of the features seen in desktop spreadsheet applications. Some of them such as Office Online, ZOHO, Google Spreadsheets, EditGrid or ZK Spreadsheet also have strong multi-user collaboration features and / or offer real time updates from remote sources such as stock prices and currency exchange rates.

A host of other currently used spread sheet software include A list of current spreadsheet software include Accel Spreadsheet from SSuite Office, Calligra Sheets (formerly KCalc),Corel Quattro Pro (WordPerfect Office),GS-Calc, GridCraft collaborative cloud spreadsheet for web and iPad, iSpread for iPad, iPhone and iPod Touch, Kingsoft Spreadsheets, Mariner Calc is Mariner Software's spreadsheet software for Mac OS, Calc XLS is Mariner Software's spreadsheet software for iOS, Numbers is Apple Inc.'s spreadsheet software, part of iWork, Quantrix Modeler, Multi-Dimensional Spreadsheet Software, ZCubes-Calc.

1.1.3. How to start with Spreadsheets?

Of the various Spreadsheet application options available, we will be using Microsoft Excel for this course. Microsoft Excel can be usually opened in a computer using the Windows Operating system by clicking Start→Programs→Microsoft Office→Microsoft Excel. When a blank Excel spreadsheet opens, it is known as a “workbook” or “book” as shown below in figure 1.

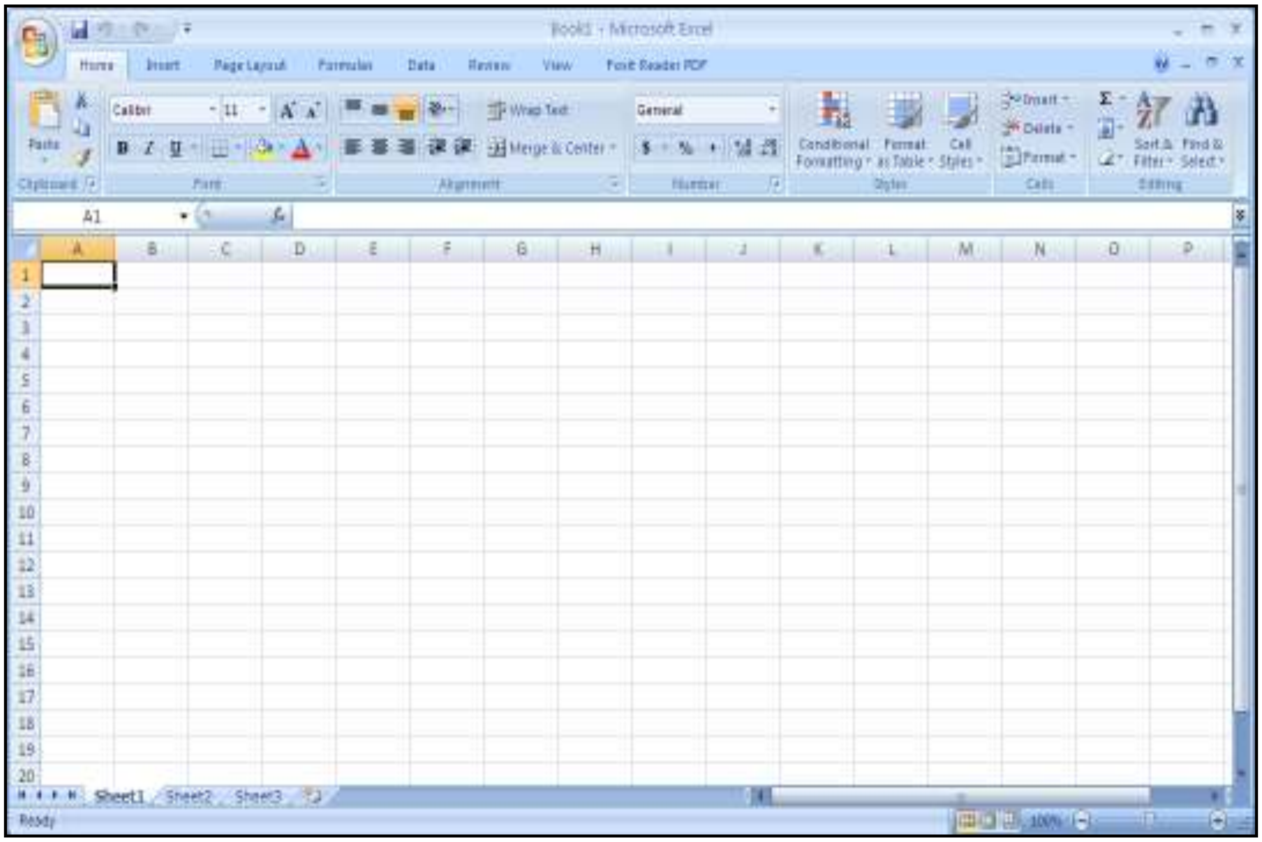


Figure 1: A work book containing 3 sheets

1.1.4. Spreadsheet Layout and Organization

The various parts of an open worksheet are shown in figure 2. A Work sheet is made of white colored rectangles called cells which are arranged in “rows” and “columns” indicated by numbers and labeled by letters respectively. A group of cells is known as a “range.” If you have multiple sheets in a book, you can quickly flip through them using the work tabs on the left side bottom. On the bottom right side, in the gray status bar, you can adjust your page views and zoom level.

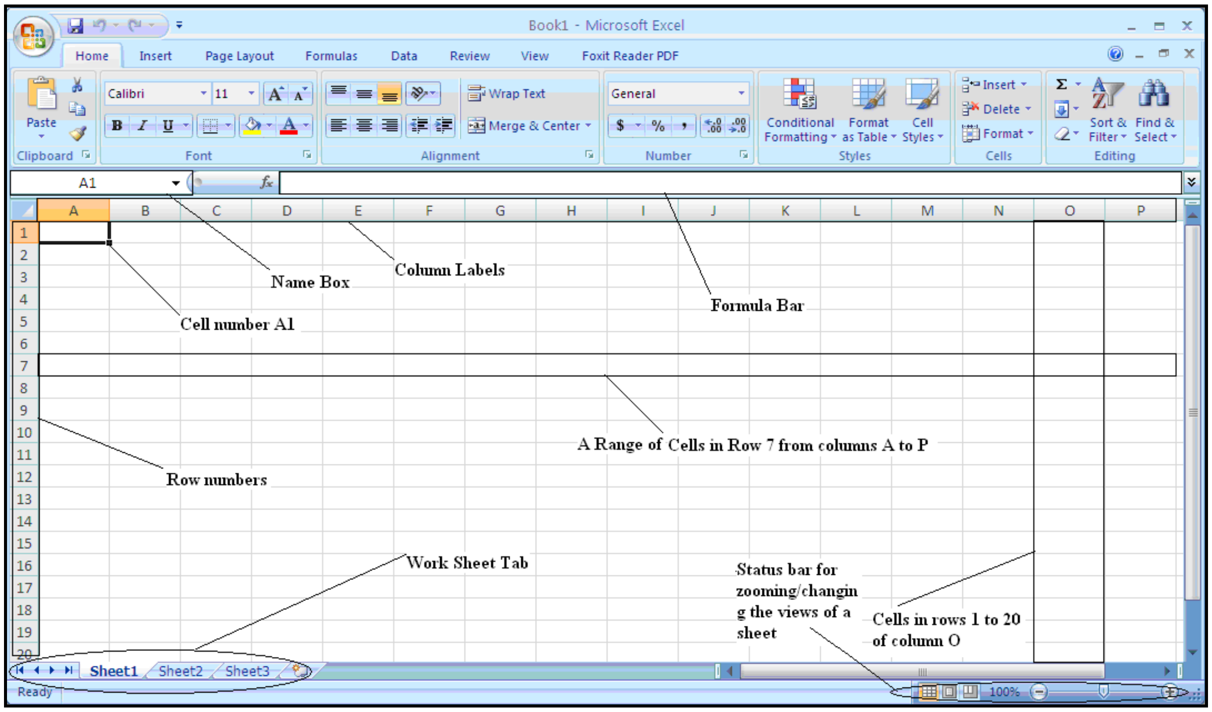


Figure 2: Parts of an Open Worksheet

The top of the Excel application is dominated by the "ribbon" (blue), which subdivided into "tabs" (red), which are then further sub-divided into "sections" (green) as shown in figure 3. The tabs are further subdivided into Home, Insert, Page Layout, Formulae, Data, Review, View etc. Each tab can be clicked which creates a drop down box called a drop-down menu which has more options which can be selected by pointing and left clicking with the mouse. Below the tabs, there are various sections such as clipboard which allows you to cut copy and paste the values. Font section allows you to change the design of the letters, their color, size, styles like underline, bold, italicize, draw an outline, fill with colors etc. The alignment section allows aligning the contents to the left or right or center inside a cell. Similarly other sections provide many options which could be explored as required.

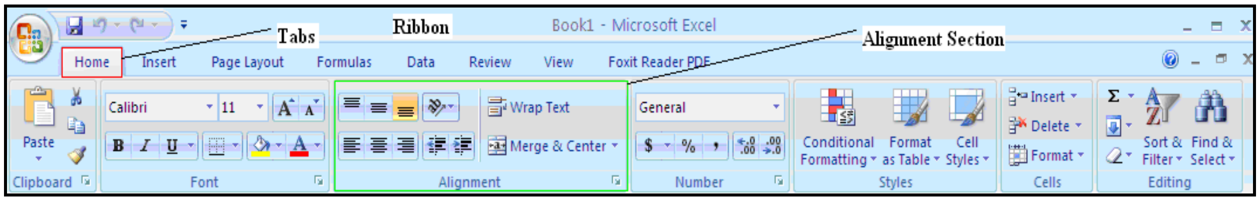


Figure 3: Ribbon, Tabs and Sections

Below the ribbon is the "name box," as shown in figure 4 which lets you rename "cells" and then to the right of that is an area that allows you to create your all-important "formulae." If you move your mouse pointer over each icon, you can see the purpose of each button, but what we're most concerned with is the wide area where our formulae and functions will be displayed.

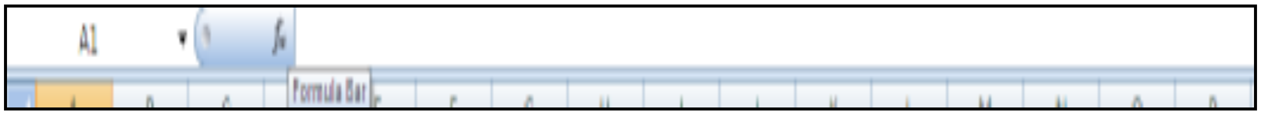


Figure 4: Name box and Formula Bar

1.1.5. Performing Built-in Calculations in Excel

A built-in calculation is an operation that takes one or more input values and outputs the result. Built in calculations can be performed by clicking any cell and typing in the calculation. All built-in calculations in Excel begin with an equal to sign ("=").For example, =AVERAGE(2,4) is a built-in calculation to find average of two input values 2 and 4 But AVERAGE(2,4) (without '=' sign) is just a string of text. Seeing the = sign, Excel knows not to treat the name average or input values as ordinary text. Without an equals sign, Excel will not calculate a result. Such built-in calculations in Excel are called as "Functions" The inputs to a function are called "arguments". For example, the arguments of the function average are 2 and 4.

Figure 5 shows how to type in the function to calculate the average of 2 numbers in excel. As the formula is typed, a small yellow box called tool tip helps you with the type of arguments that must be given to the function.

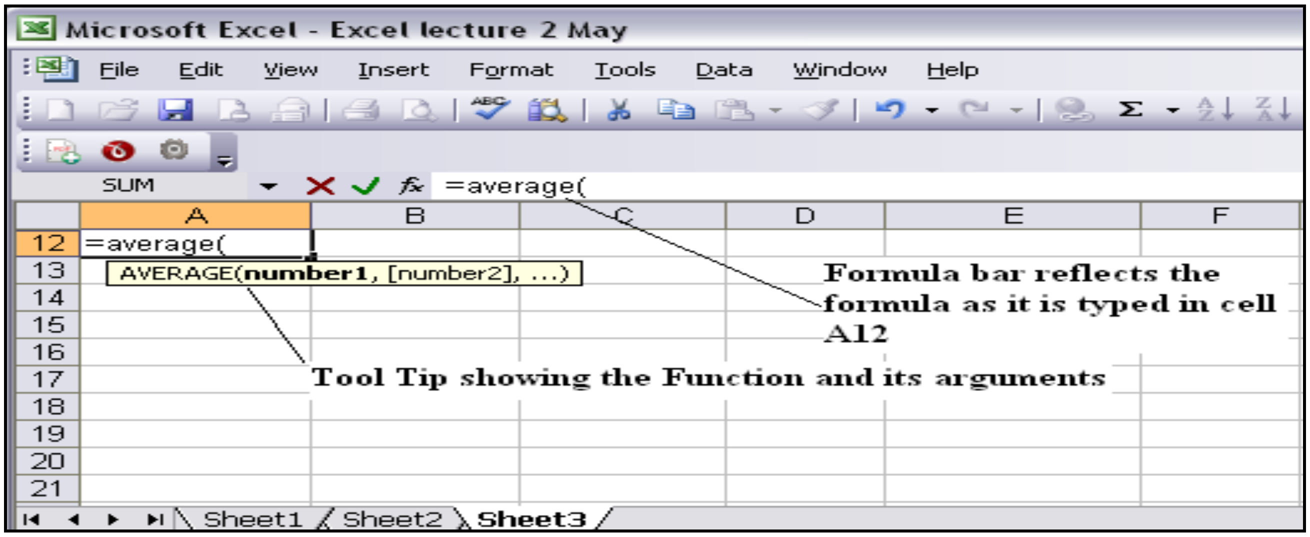

Microsoft Excel - Excel lecture 2 May
File
Edit
View
Insert
Format
Tools
Data
Window
Help
SUM
=average(
A
B
C
D
E
F
12
13
14
15
16
17
18
19
20
21
=average(
AVERAGE(number1, [number2], ...)
Formula bar reflects the formula as it is typed in cell A12
Tool Tip showing the Function and its arguments
Sheet1
Sheet2
Sheet3

Figure 5: Typing the average formula in worksheet 3, Cell A12

The left worksheet in Figure 6 shows the complete formula typed in cell A12. After typing the formula, if the enter key is pressed, the result is displayed if the result is correct. Other wise an error shows up as a message box which needs to be corrected as shown in figure 7. The worksheet on the right of figure 6 shows the result displayed as a result of the execution of the average formula in A12.

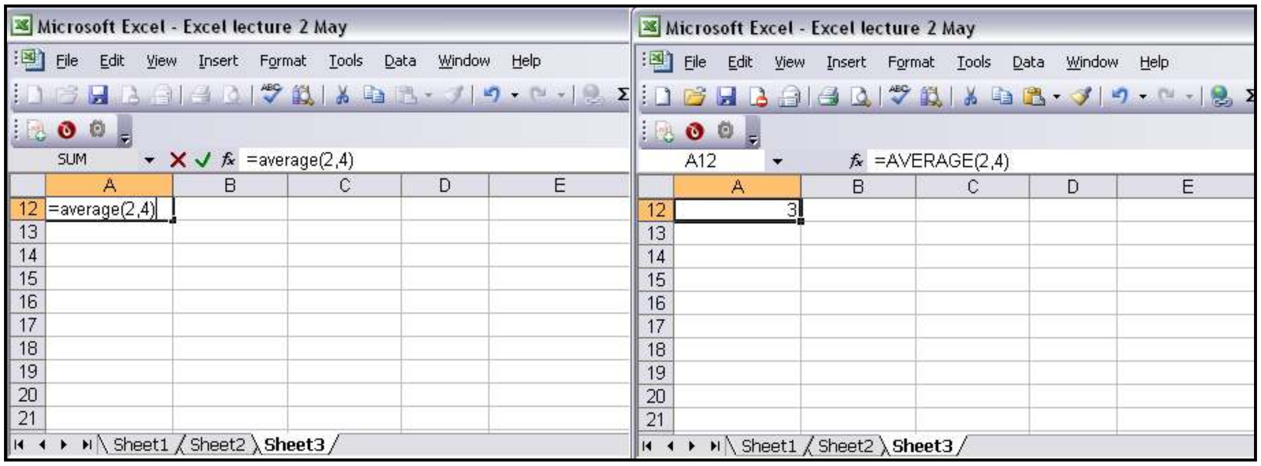

Microsoft Excel - Excel lecture 2 May
File Edit View Insert Format Tools Data Window Help
SUM
=average(2,4)
A12
=AVERAGE(2,4)
3
Sheet1 Sheet2 Sheet3

Figure 6: Results of executing the average formula

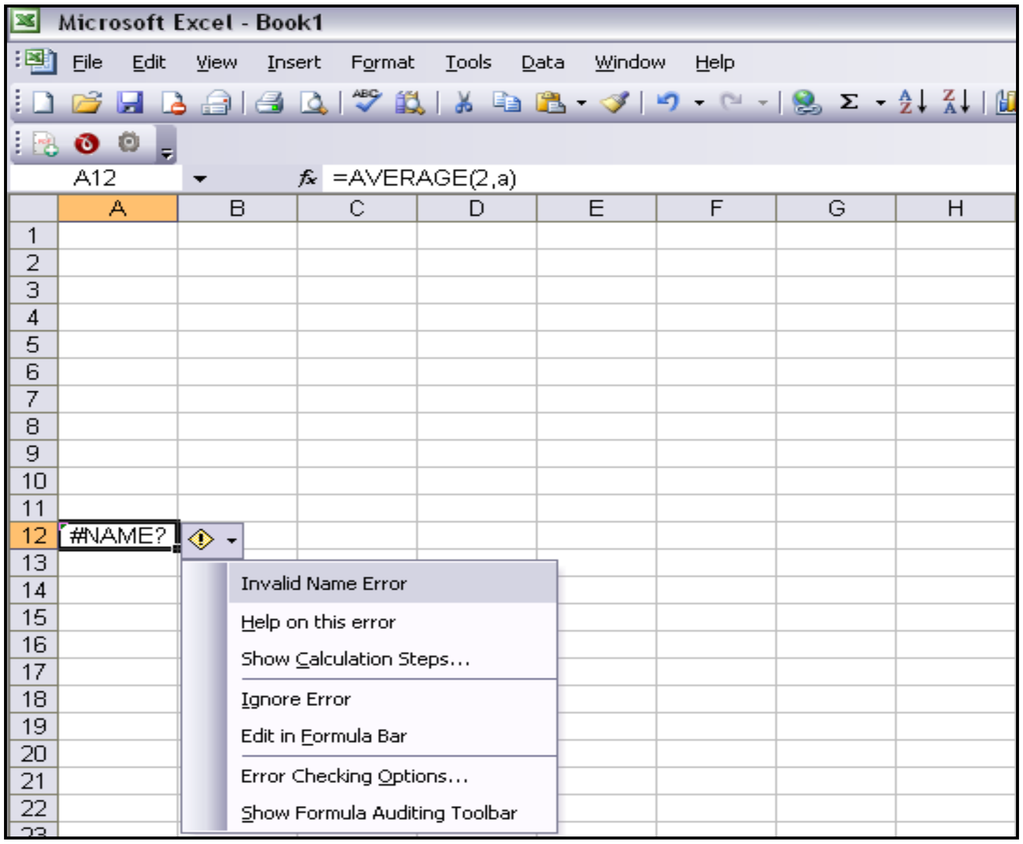

Microsoft Excel - Book1
File Edit View Insert Format Tools Data Window Help
A12
=AVERAGE(2,a)
#NAME?
Invalid Name Error
Help on this error
Show Calculation Steps...
Ignore Error
Edit in Formula Bar
Error Checking Options...
Show Formula Auditing Toolbar

Figure 7: Result of executing a wrong formula to find average of a number and an alphabet

Note: Excel uses upper-case (capital) letters to list functions, but you can use lower - case (small) or upper-case letters when you write them.

1.1.6. Function library in Excel

In Excel 2007, the "Function Library" can be found on the "Formulae" tab as shown in figure 8 or it could be obtained by clicking Insert → Function in older versions of Microsoft Excel as shown in figure 9. The function library consists of a host of built-in functions which are broadly classified into Add-in and Automation functions, Cube functions, Database functions, Date and time functions, Engineering functions, Financial functions, Information functions, Logical functions, Lookup and reference functions, Math and trigonometry functions, Statistical functions, Text functions etc. For example average is an example of a mathematical function. The usage of these built in functions is similar to the average function example discussed in the previous section 1.1.5.

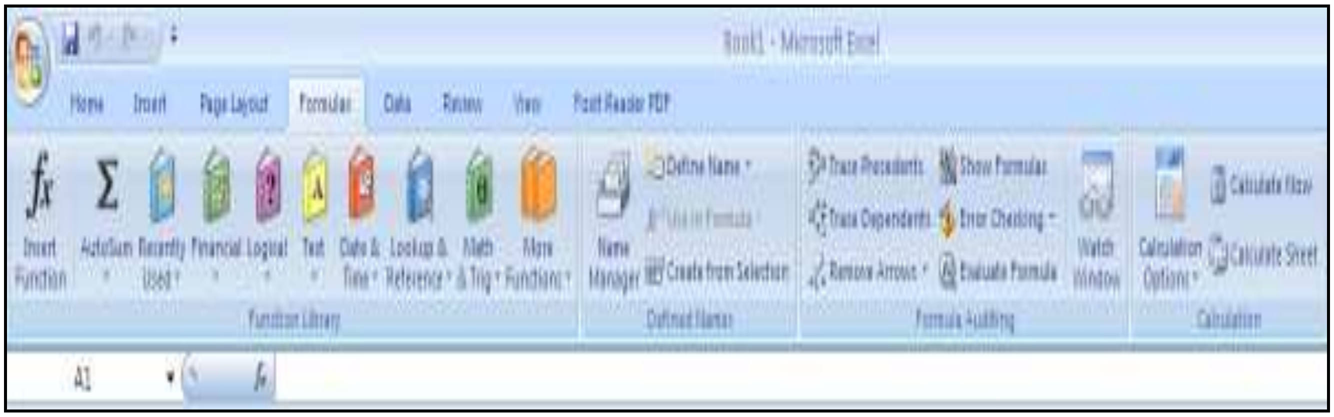
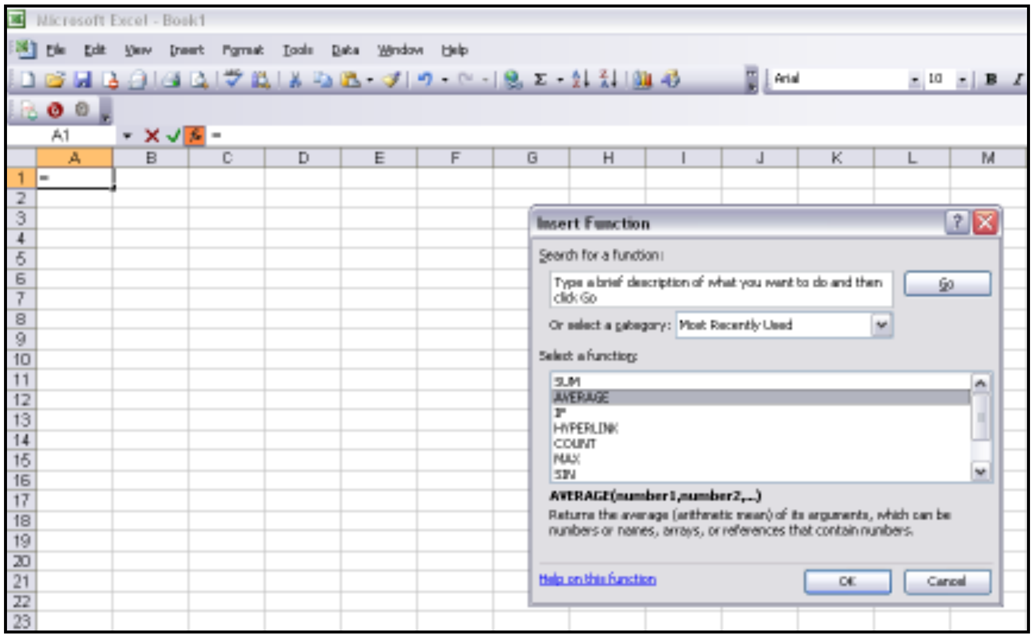


Figure 8: Formula Library in Microsoft Excel 2007



Search for a function:

Type a brief description of what you want to do and then click Go

Or select a category: Most Recently Used

Select a function:

SUM
AVERAGE
IF
HYPERLINK
COUNT
MAX
SIN

AVERAGE(number1,number2,...)

Returns the average (arithmetic mean) of its arguments, which can be numbers or names, arrays, or references that contain numbers.

Figure 9: Insert Function Library in Microsoft Excel 2007

1.1.7. Formulae, Names, Addresses and Operators in Microsoft Excel

A formula in Excel is combination of "operators", "operands", and "functions." For example, the function =SUM adds a list of numbers (it is so commonly used, that is listed on the first menu in Excel, abbreviated by the Greek letter Sigma (Σ), which is the notation that mathematicians use to sum a series). A formula is used similar to performing a calculation by hand. For example, family budget calculation can be written as shown below:

Remaining cash = (4 * weekly salary) – mortgage – food – utilities

The symbols '* 'And '–'are called *"operators"* and are used to perform multiplication and division respectively. Table 1 shows a list of common mathematical operators used for performing general arithmetic on numbers and Table2 shows a list of operators used generally for comparing the values in two cells which may result in either TRUE or FALSE results. The values 'weekly salary', 'mortgage', 'food' and 'utilities' are known as *"operands"*. The result of executing this formula is the value named as 'remaining cash'.

The values for "food" and the other operands are names that you define in Excel. Without a "name," you would have to use the "address." The address of a cell is written using row-column notation. The rows are given numbers and the columns, letters. The first cell in the spreadsheet is A1.When you reached the end of the alphabet, the rows are numbered AA, AB, BA, BB, etc.

Table 1: Common Mathematical Operators in Microsoft Excel

Math Operator	Definition
+	Addition
-	Subtraction, or negation, e.g., 6 * -1 = -6
*	Multiplication
/	Division
%	Percentage

Table 2: Common Comparison Operators in Microsoft Excel

Comparison Operator	Definition
=	Equals, e.g., 2=4 or "b" = "b"
>	Greater than, e.g., 4 > 2 or "b" > "a"
<	Less than, e.g., 2 < 4 or "a" < "b"
>=	Greater than or equal to – another way to think of this is >= means *either* > *or* =.
<=	Less than or equal to.
<>	Not equal to, e.g., 4<>6

1.1.8. Operator Order Precedence in Microsoft Excel

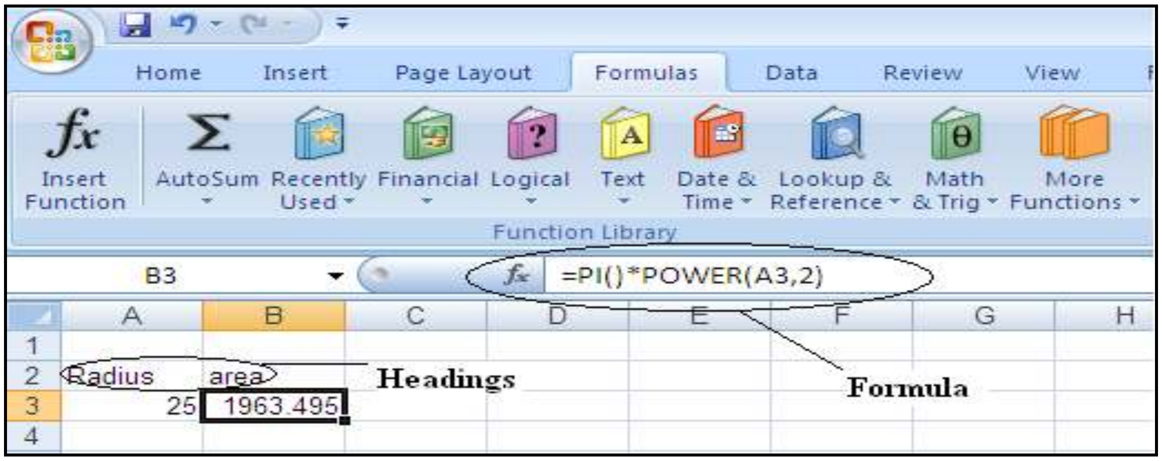
Excel first evaluates items in parentheses working inside out. It then uses the order precedence rules of mathematics. When two items have the same precedence, Excel works left to right. The precedence of math operators is shown below in table 3, in descending order.

Table 3: Operator Precedence of Excel Operators in Descending Order

(and)	When parentheses are used, they override the normal rules of precedence. This means that Excel will do this calculation first.
-	Negation, e.g., -1. This is the same as multiplying a number by -1. -4 = 4 * (-1)
%	Percent, means multiply by 100. E.g., 0.003 = 0.3%.
power()	Exponentiation
* and /	Multiply and divide. How can two operators have the same precedence? It just means that if a formula has two more operators with the same precedence, then the calculation is done left to right.
+ and -	Addition and subtraction.

1.1.9. Simple Examples of Formulae usage in Microsoft Excel

An example to illustrate the usage of formula is shown in figure 10 which calculates the area of a circle. The Radius of a circle is entered in cell A3 and the formula for area is typed in cell B3 which gets displayed in the formula box as you type the formula in B3. The result appears in B3 after the enter key is pressed. When the cell B3 is clicked with left mouse button the corresponding formula will get displayed in the formula box. For the user's reference, a heading called Radius and Area can be typed into cells for identification. Here the heading is typed in the cells A2 and B2 to show that the row below A2 and B2 (3rd row) contains the radius value and area value respectively. One can see that the formula in the formula box uses functions called PI and Power which are present in the function library. This justifies the definition of a formula as given in the beginning of section 1.1.7

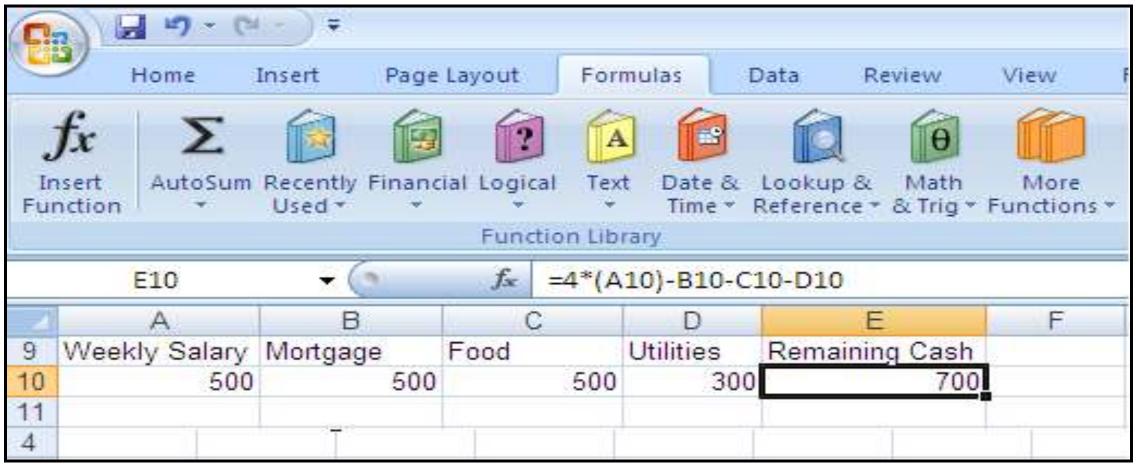


=PI()*POWER(A3,2)

	A	B
1		
2	Radius	area
3	25	1963.495
4		

Figure 10: A simple example using formula to calculate the area of a circle given the radius

Figure 11 shows the calculation of Family budget example discussed in section 1.1.7 using Excel. One can see that the formula refers to the cells (A10, B10, C10 and D10) to calculate the results and not the headers. The headings are just for the user's reference to identify what is what!



E10 =4*(A10)-B10-C10-D10

	A	B	C	D	E	F
9	Weekly Salary	Mortgage	Food	Utilities	Remaining Cash	
10	500	500	500	300	700	
11						
4						

Figure 11: Example - Family budget calculation

1.1.10. Copying a Formula to a Range of Cells

Excel allows that a formula can be written in a cell and copied into to a large area and the reference gets automatically updated. This avoids having to edit each cell to ensure it points to the correct place. ”Range” means more than one cell. For example, (C1:C10) means all the cells from cell C1 to cell C10 (a column of cells).If a range cross five columns and ten rows, then you indicate the range by writing the top-left cell and bottom right one, e.g., A8:E18. This is a square area that crosses rows and columns and not just part of a column or part of a row as shown in figure 12.

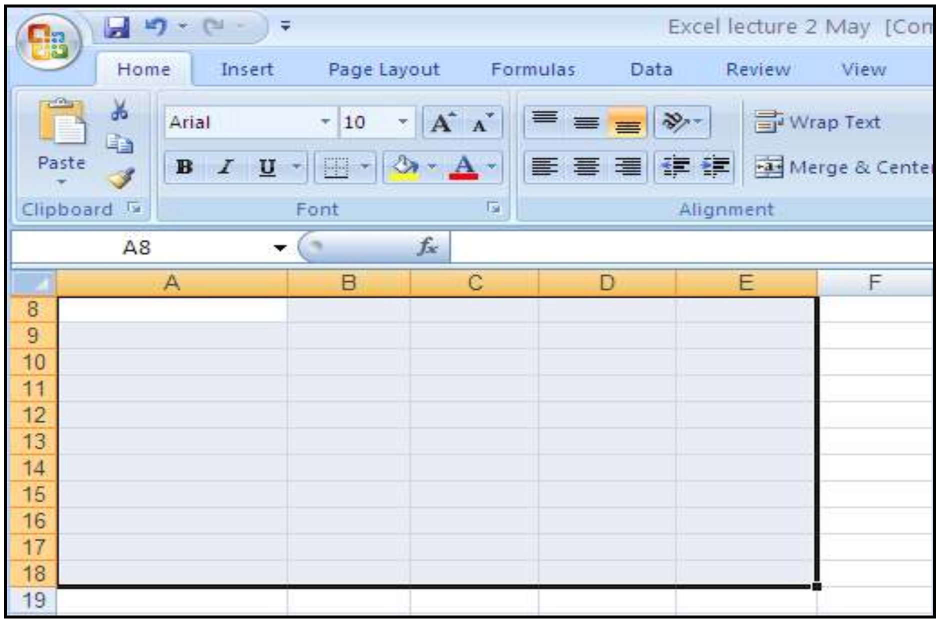
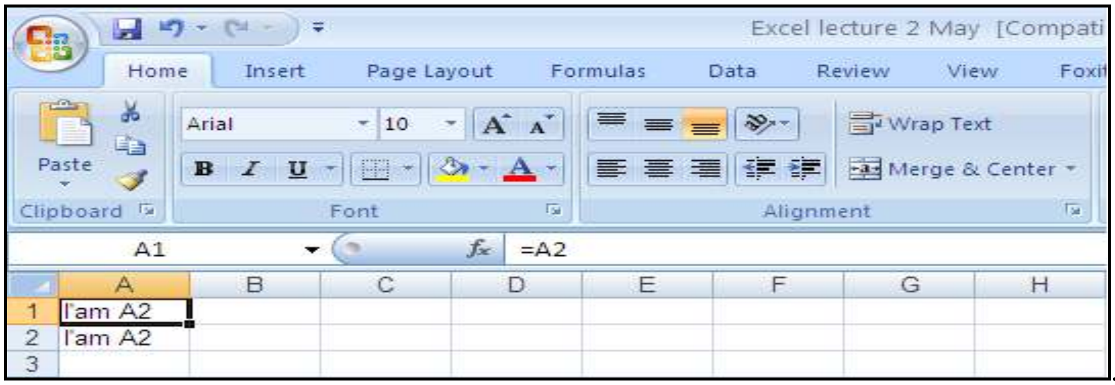


Figure 12: A square range of cells

1.1.11. Cell Reference

A “cell reference” means the cell to which another cell refers. For example, if in cell A1 you have =A2. Then A1 refers to A2 as shown in figure 13.



A1 =A2

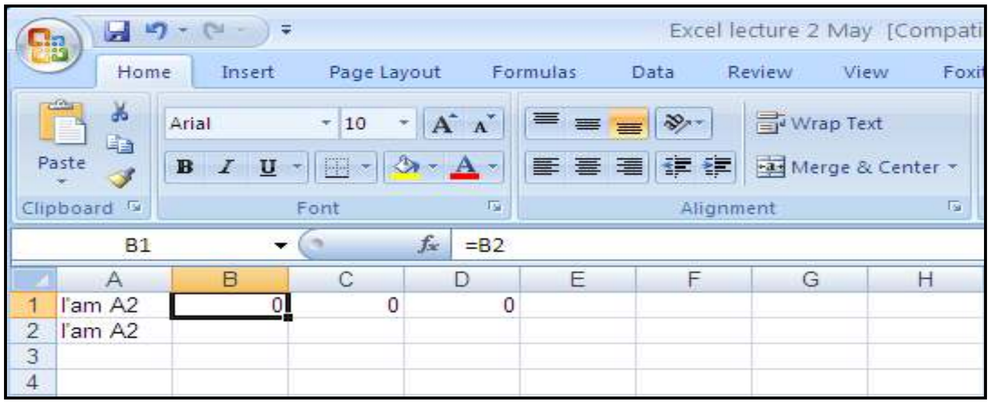
	A	B	C	D	E	F	G	H
1	I'am A2							
2	I'am A2							
3								

Figure 13: Cell Reference

1.1.12. Types of Cell Reference

There are three types of cell references

- **Relative** – Relative referencing means that the cell address changes as you copy or move it; i.e. the cell reference is relative to its location. Referring the earlier example in figure 13, suppose in cell A1 we have a formula that simply says =A2. That means Excel output in cell A1 whatever is inputted into cell A2. In cell A2 we have typed "I'am A2" so Excel displays the value "I'am A2" in cell A1 as shown in figure 13. As shown in figure 14, if you copy the contents of cell A1 to B1, the formula now refers to the cell B2 automatically i.e., it assumes that if A1=A2, B1=B2. B1 is zero as B2 does not contain anything. This is shown in figure 14.

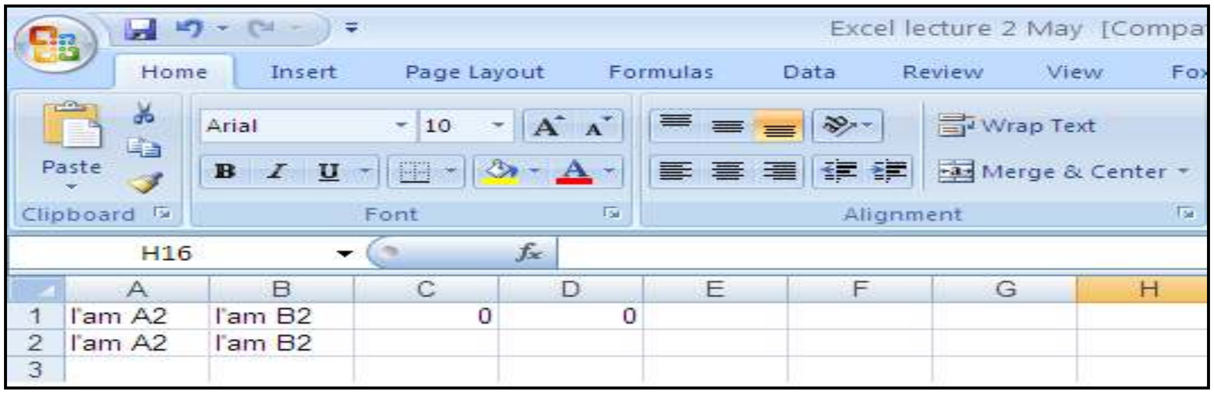


B1 =B2

	A	B	C	D	E	F	G	H
1	I'am A2	0	0	0				
2	I'am A2							
3								
4								

Figure 14: Relative Reference

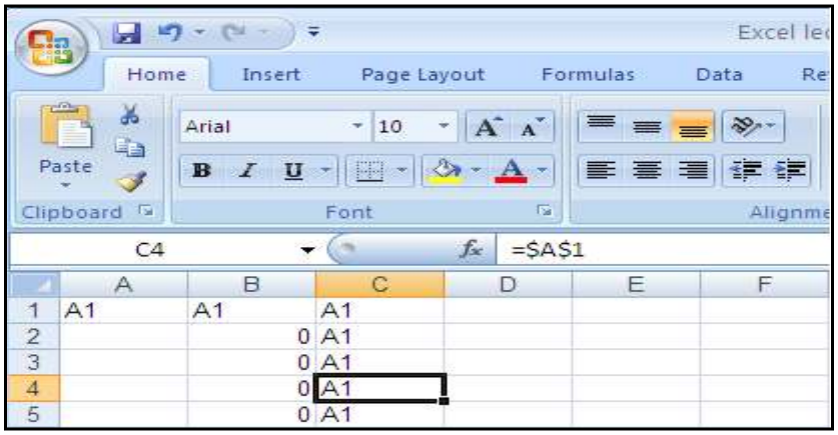
Now, Typing I'am B2 into the cell B2 changes the contents of the cell B1 to I'am B2 as shown in figure 15.



	A	B	C	D	E	F	G	H
1	I'am A2	I'am B2	0	0				
2	I'am A2	I'am B2						
3								

Figure 15: Modified Relative Reference

- **Absolute** – This means the cell reference stays the same if you copy or move the cell to any other cell. This is done by anchoring the row and column, so it does not change when copied or moved. The $ sign is used to make an absolute reference (anchor).For example, enter the formula =$A$1 in any cell. The $ in front of the column A means do not change the column, the $ in front of the row 1 means do not change the column when you copy or move the cell to any other cell as shown in figure 16.



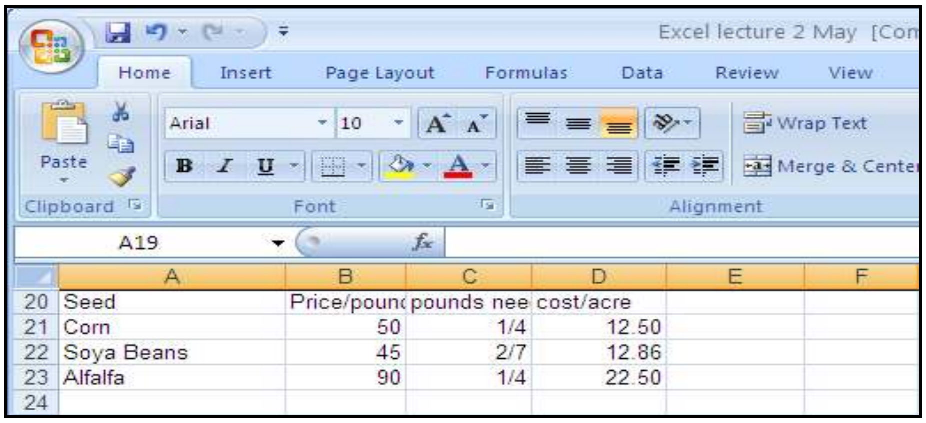
=$A$1

	A	B	C	D	E	F
1	A1	A1	A1			
2		0	A1			
3		0	A1			
4		0	A1			
5		0	A1			

Figure 16: Absolute Reference (C-Column) Vs Relative Reference (B-Column)

- **Mixed** – This means you can choose to anchor either the row or the column when you copy or move the cell, so that one changes and the other does not. For example, you could anchor the row reference then move a cell down two rows and across four columns and the row reference stays the same. Mixed references are when either the row or column is anchored.

For example, suppose you are a farmer making a budget. You also own a feed store and sell seeds. You are going to plant corn, soybeans, and alfalfa. The spreadsheet below shows the cost per acre. The "cost per acre" = "price per pound" * "pounds of seeds per acre" – that's what it will cost you to plant an acre. Enter the cost per acre as =$B21 * C21 in cell D21. You are saying you want to anchor the price per pound column. Then copy that formula to the other rows in the same column as shown below in figure 17.This applies the same formula to all other rows which is the concept of iteration in Excel.

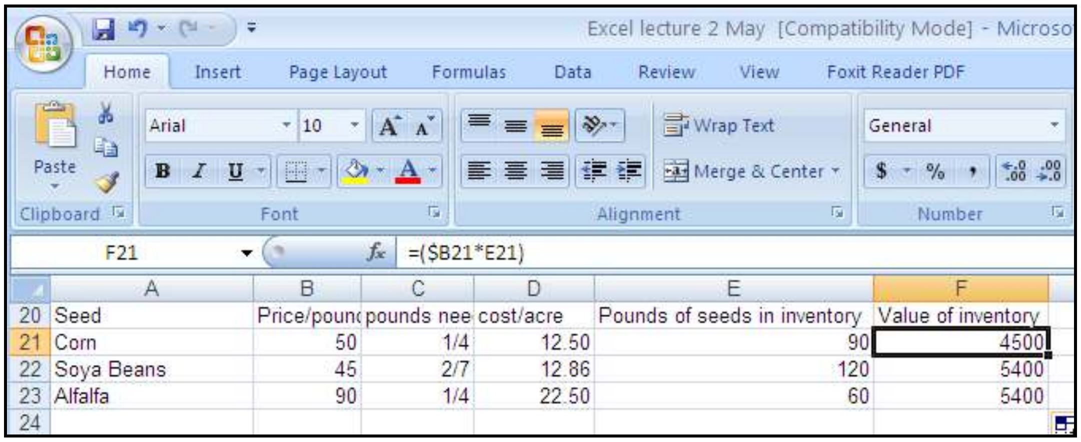


A19

	A	B	C	D	E	F
20	Seed	Price/pounc	pounds nee	cost/acre		
21	Corn	50	1/4	12.50		
22	Soya Beans	45	2/7	12.86		
23	Alfalfa	90	1/4	22.50		
24						

Figure 17: Mixed Reference Example – Data Entry

Now you want to know the value of your inventory of seeds. You need the price per pound and the number of pounds in inventory to know the value of the inventory. We add two columns: “pound of seed in inventory” and then “value of inventory.” Now, copy the cell D21 to F23 and note that the row reference in the first part of the original formula ($B21) is updated to row 4 but the column remains fixed because the $ anchors it to “B.” This is a mixed reference because the column is absolute and the row is relative. The excel calculation is shown in figure 18.

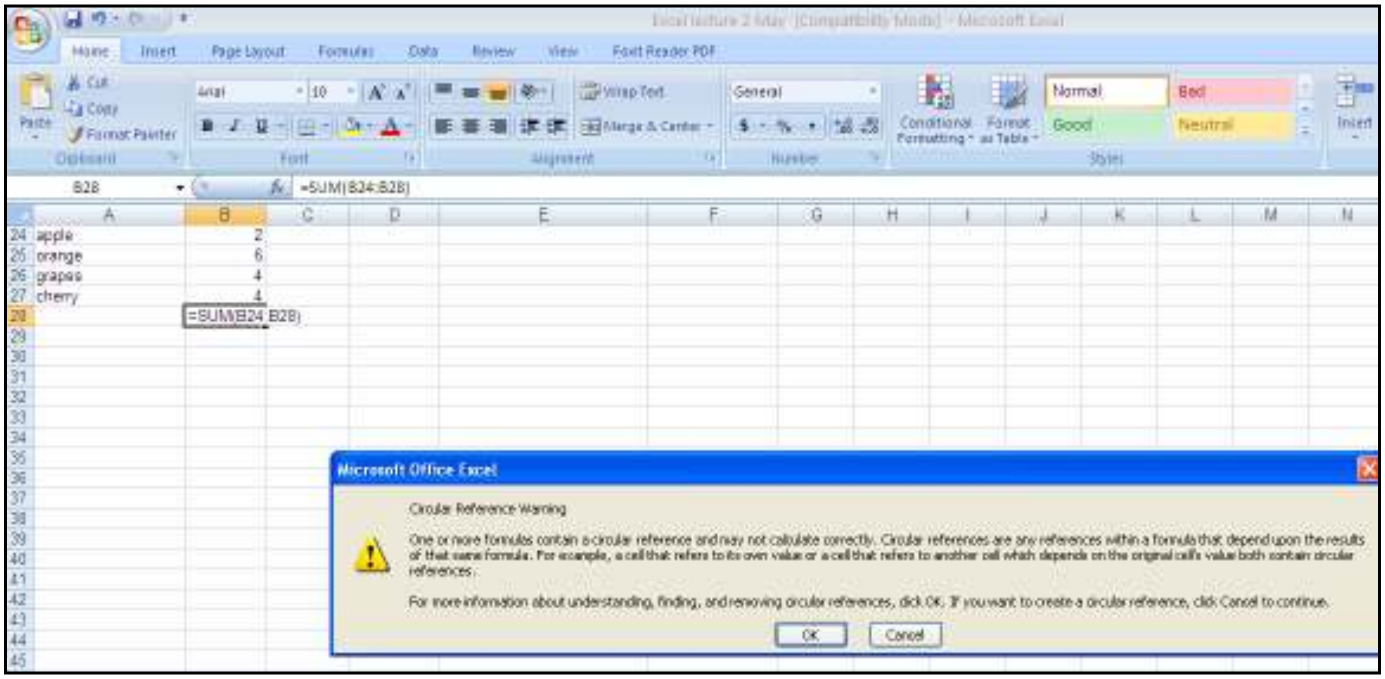


F21 =($B21*E21)

	A	B	C	D	E	F
20	Seed	Price/pounc	pounds nee	cost/acre	Pounds of seeds in inventory	Value of inventory
21	Corn	50	1/4	12.50	90	4500
22	Soya Beans	45	2/7	12.86	120	5400
23	Alfalfa	90	1/4	22.50	60	5400
24						

Figure 18: Mixed Reference Example - Calculations

- **Circular Reference -** A circular reference is when a formula refers to itself. For example, you cannot write c3 = c3 + 1. This kind of calculation is called “iteration” meaning it repeats itself. Excel does not support iteration because it calculates everything only one time. If you try do this by typing SUM(B24:B28) in cell B28 an error is shown as in the below figure 19.


B28 =SUM(B24:B28)

	A	B
24	apple	2
25	orange	6
26	grapes	4
27	cherry	4
28		=SUM(B24:B28)

Microsoft Office Excel

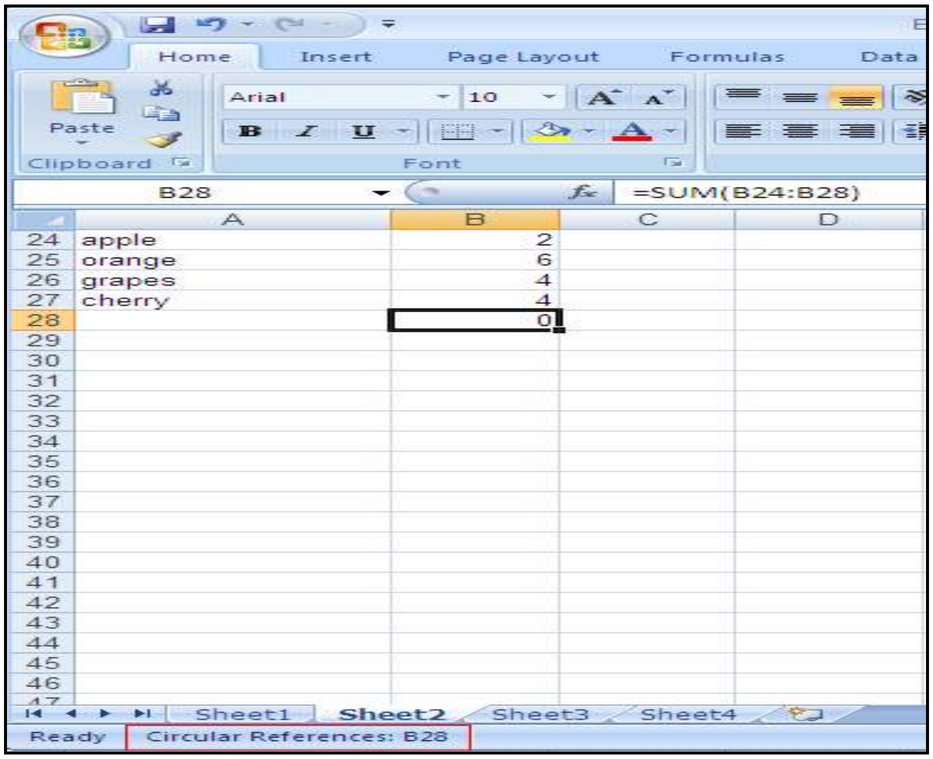
Circular Reference Warning

One or more formulas contain a circular reference and may not calculate correctly. Circular references are any references within a formula that depend upon the results of that same formula. For example, a cell that refers to its own value or a cell that refers to another cell which depends on the original cell's value both contain circular references.

For more information about understanding, finding, and removing circular references, click OK. If you want to create a circular reference, click Cancel to continue.

Figure 19: Circular Reference

Excel only tells you that you have a circular reference at the bottom of the screen so you might not notice it. If you do have a circular reference and close a spreadsheet and open it again, Excel will tell you in a pop-up window that you have a circular reference. If you do have a circular reference, every time you open the spreadsheet, Excel will tell you with that pop-up window that you have a circular reference as shown in figure 20.


B28 =SUM(B24:B28)

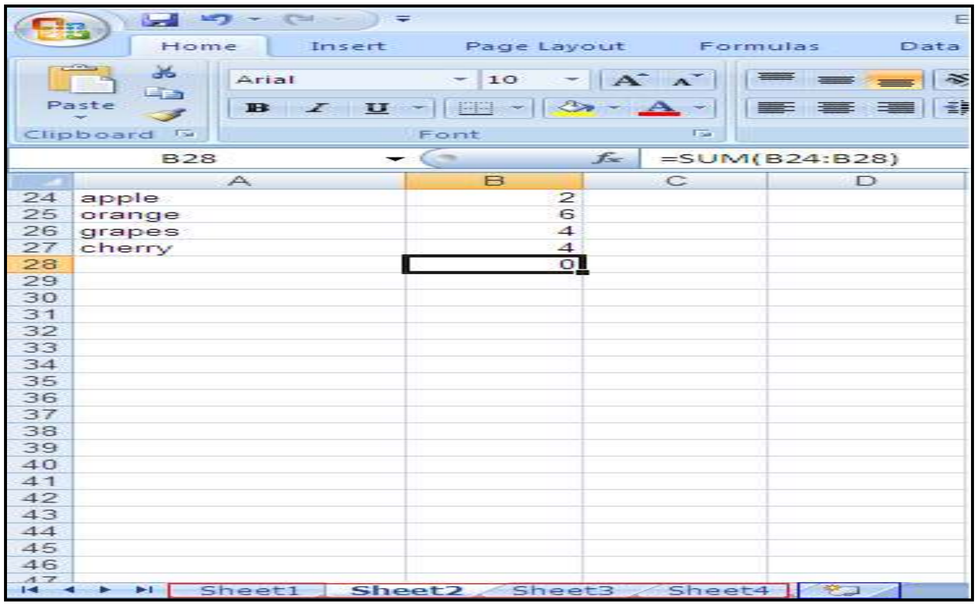
	A	B
24	apple	2
25	orange	6
26	grapes	4
27	cherry	4
28		0

Ready Circular References: B28

Figure 20: Circular Reference indication at the bottom of the window

1.1.13. Working with Multiple Worksheets

A “workbook” is a collection of “worksheets.” Simply put, this means you can have multiple spreadsheets (worksheets) in the same Excel file (workbook). As you can see in the figure 21 below, our example workbook has many worksheets (in red).Worksheets by default are named Sheet1, Sheet2, and so forth. You create a new one by clicking the small sheet (blue) at the bottom of the Excel screen or by using insert → worksheet as in older versions of excel.



	A	B
24	apple	2
25	orange	6
26	grapes	4
27	cherry	4
28		0

Figure 21: Inserting a new worksheet

You can change the worksheet name to something useful like “loan” or “budget” by right-clicking on the worksheet tab shown at the bottom of the Excel program screen, selecting rename, and typing in a new name. Or you can simply double-click on the tab and rename it as shown in figure 22.

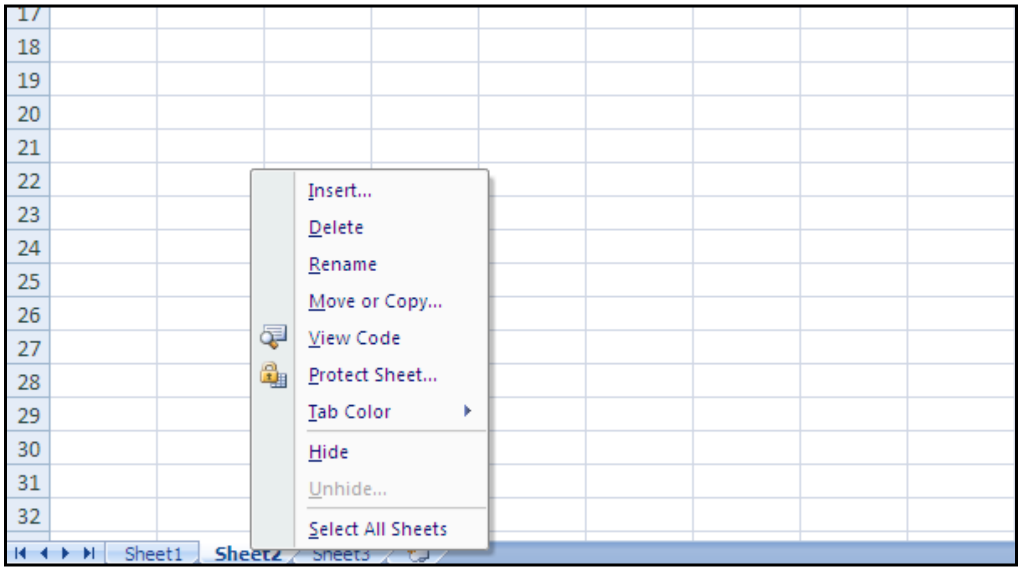


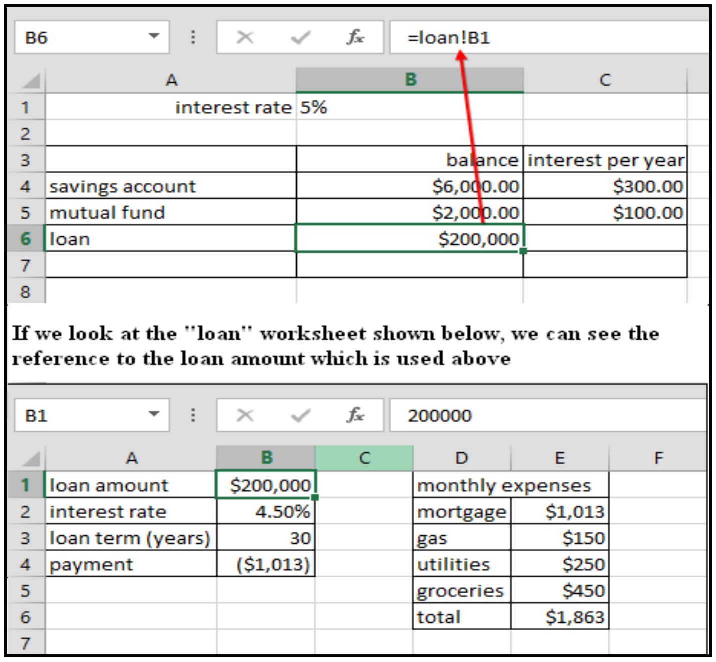
Figure 22: Renaming a worksheet

1.1.14. References to Other Worksheets

The formula for a worksheet reference is =worksheet!cell. You can use this kind of reference when the same value is used in two worksheets, examples of that might be:

- Today's date
- Currency conversion rate from Dollars to Euros
- Anything that is relevant to all the worksheets in the workbook

- Below in figure 23 is an example of worksheet "interest" making reference to worksheet "loan," cell B1.



B6 =loan!B1

	A	B	C
1	interest rate	5%	
2			
3		balance	interest per year
4	savings account	$6,000.00	$300.00
5	mutual fund	$2,000.00	$100.00
6	loan	$200,000	
7			
8			

If we look at the "loan" worksheet shown below, we can see the reference to the loan amount which is used above

B1 200000

	A	B	C	D	E	F
1	loan amount	$200,000		monthly expenses		
2	interest rate	4.50%		mortgage	$1,013	
3	loan term (years)	30		gas	$150	
4	payment	($1,013)		utilities	$250	
5				groceries	$450	
6				total	$1,863	
7						

Figure 23: Referencing a worksheet - Example

1.1.15. Useful Functions - Math and Financial Functions

Table 4: Some commonly used Mathematical and Financial Functions in Microsoft Excel

Functions	Purpose
SQRT	Square root, e.g., SQRT(4)=2 because 2*2=4
PMT	Loan payment
DEGREES	Used by engineers, for example, to convert degrees (e.g., 360□) to radians (e.g. 2π)
GCD	Finds the greatest-common divisor between two numbers. For example, GDC(5,15)=5, because 5 is the largest number that divides 15. 3 also divides 15, but that is not the *largest* divisor they have in common.
RAND(23)	Generate a random decimal number between or equal to 0 and 1. You could use this for example, to pick a contest winner (see example below). RANDBETWEEN is easier to use, since you do not have convert this decimal number to a whole number, e.g., 3 is easier to use than 0.3.

1.1.16. Useful Functions - Logical Functions

Logical functions are used to test whether something is true or false.

Table 5: Some commonly used Logical Functions in Microsoft Excel

Function	Purpose
IF	If(<test>,X,Y)If <test> is true, then the value is X, otherwise the value is Y. For example this formula =IF(C2=5,"C2=5″,"C2<>5″)Displays "C2=5" if C2=5, otherwise it will display "C2<>5"
AND	=AND(C3=5,C4=5)Displays "TRUE" if both C3=5 and C4=5
FALSE	Sets the cell to false
TRUE	Sets the cell to true

1.1.17. Useful Functions - Date and Time Functions

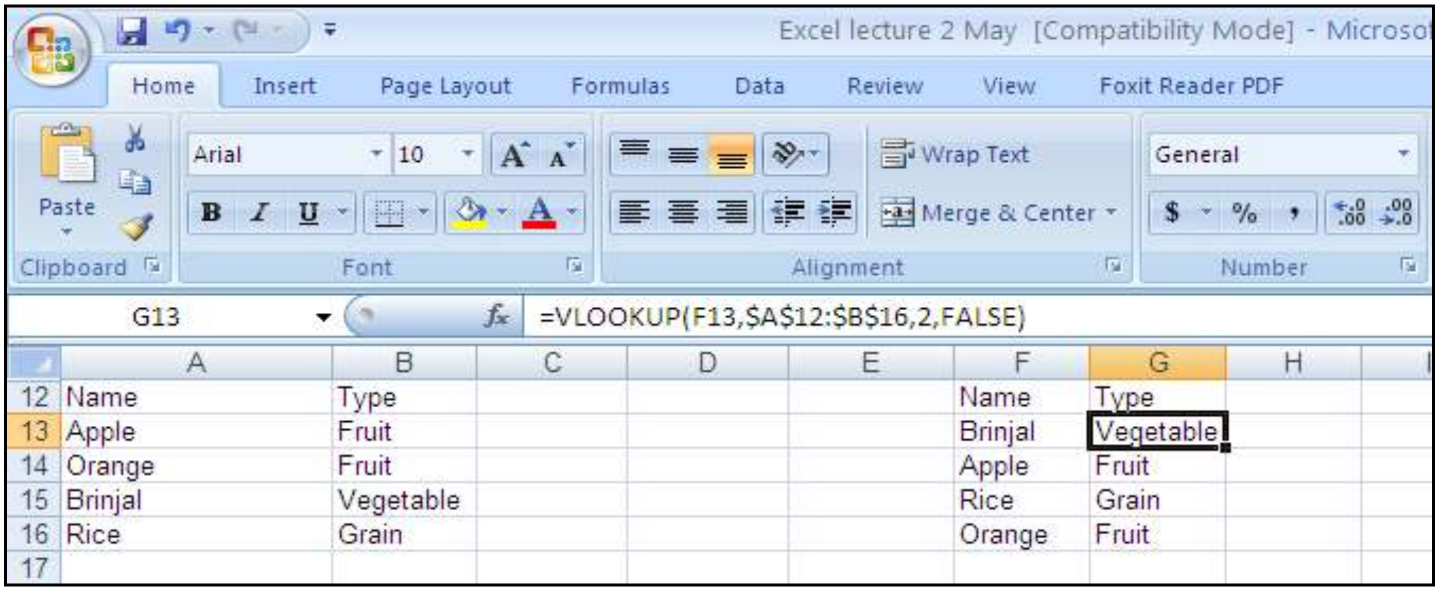
Dates are stored in excel as numbers, which means you can do math with them. You can use +1 to calculate the next day and you can subtract one date from another to find how many days have elapsed between two dates

Table 6: Some commonly used Date and Time Functions

Formula	Result	Purpose
=NOW()	9/16/2013	Get the current date and time
=DATEVALUE("9/16/2013″)	41533	Convert date in text format to number. This number is called the "serial number". Number 1 is January 1, 1900. 41,533 is 41,532 days after that or 9/16/2013.
=WEEKDAY(NOW())	Number of the current day when the formula is executed	Extracts the date as a number. 1 is Sunday and 7 is Saturday.
=NOW()+1	9/17/2013	Since dates in Excel are stored as numbers, you can do math with them.

1.1.18. VLOOKUP and HLOOKUP

Vertical lookup (VLOOKUP) and horizontal lookup (HLOOKUP) functions are used to translate a number or other value into something which is understandable For example, you can use VLOOKUP to take a name and return the type =vlookup(lookup value, table where values reside, column # where values are located, false). In the below figure 24, the cells from A12 to B16 form a table of names and types of fruits and vegetables. Column F has a list of names in some order. Using VLOOKUP in column G, The table in A12 to B16 is referred to fetch the type of each name. This is very useful function when the list is very big and manual inspection and updating becomes tedious.

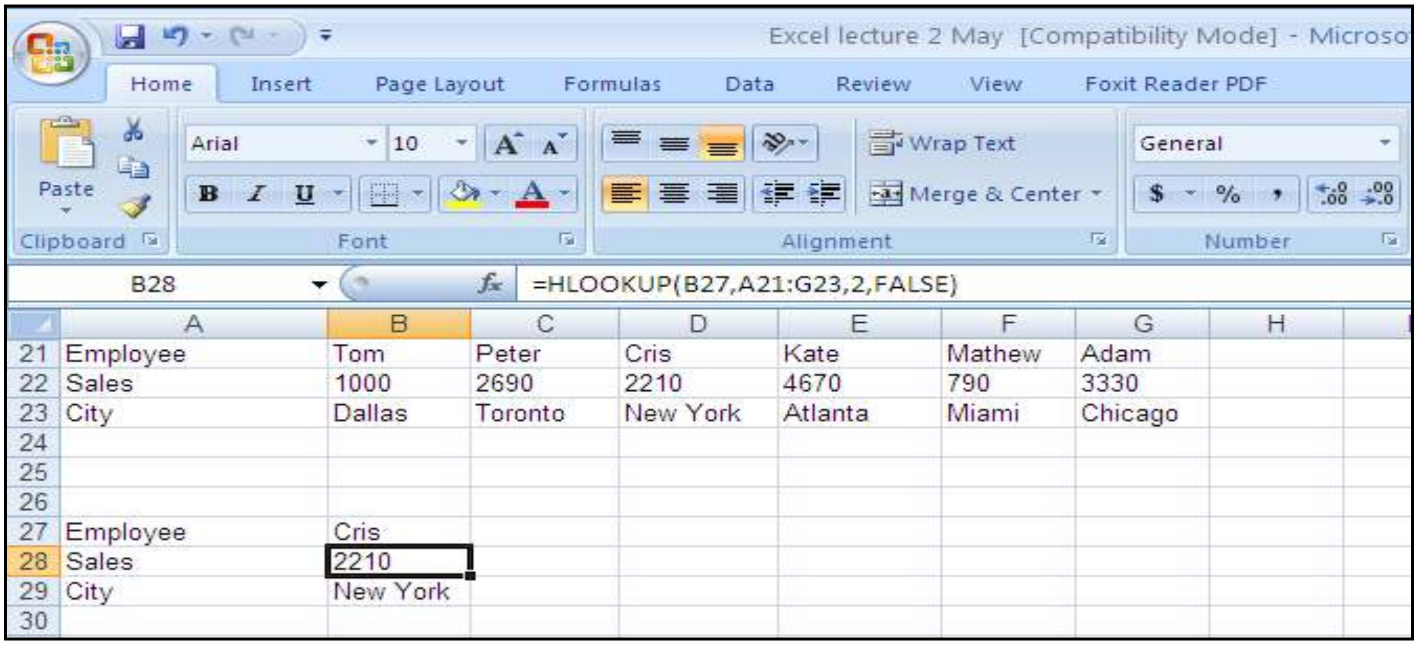


G13 =VLOOKUP(F13,$A$12:$B$16,2,FALSE)

	A	B	C	D	E	F	G	H
12	Name	Type				Name	Type	
13	Apple	Fruit				Brinjal	Vegetable	
14	Orange	Fruit				Apple	Fruit	
15	Brinjal	Vegetable				Rice	Grain	
16	Rice	Grain				Orange	Fruit	
17								

Figure 24: VLOOKUP - Example

HLOOKUP is another lookup and reference function which function retrieves data from the table horizontally in contrast to VLOOKUP. Most of the searches in Excel, are made vertically, so this feature is rarely used =HLOOKUP(reference,array,row_number,row). In the below figure 25, the cells from A21 to G23 form a table of employees with sales in different cities. To find the sales made by Cris in B28, The HLOOKUP formula is typed in B28 which gives the sales obtained from the table. Similarly the city name also can be fetched from the table using the HLOOKUP formula: =HLOOKUP(B27,A21:G23,3,FALSE). If you drag the formula from B28 then automatically the city name gets updated in B29 due to relative referencing.



B28 =HLOOKUP(B27,A21:G23,2,FALSE)

	A	B	C	D	E	F	G	H
21	Employee	Tom	Peter	Cris	Kate	Mathew	Adam	
22	Sales	1000	2690	2210	4670	790	3330	
23	City	Dallas	Toronto	New York	Atlanta	Miami	Chicago	
24								
25								
26								
27	Employee	Cris						
28	Sales	2210						
29	City	New York						
30								

Figure 25: HLOOKUP – Example

1.2. Scratch

1.2.1. An Introduction to Scratch

Scratch is a programming language developed by the MIT Media Lab for the purpose of teaching programming to teens and other first-time programmers. Scratch is a new programming language, initially released in May 2007. Scratch supports the development of computer games, interactive stories, graphic artwork and computer animation, and all sorts of other multimedia projects.

Scratch allows new programmers to create programs by snapping together blocks. Scratch consists of a programming language made up of different blocks and an easy to learn graphical development environment that includes a paint application for creating graphics and built-in sound editing capabilities. It also comes with huge collections of sample applications as well as graphics and sound files, all of which you can use to create your own Scratch projects.

As demonstrated in Figure 1, Scratch programs are made up of graphical blocks, which are snapped together. Scratch blocks resemble puzzle pieces in the way that they snap together. Scratch blocks can only be snapped together in ways that make sense, preventing new programmers from using them in invalid combinations. In this way, Scratch enforces proper programming syntax and ensures that new programmers learn the proper way to assemble and formulate programming logic.

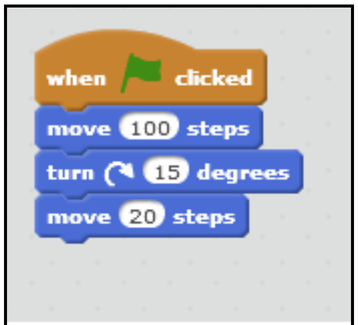

```
when [flag] clicked
move 100 steps
turn ↻ 15 degrees
move 20 steps
```

Figure 1: An example of Scratch scripts

1.2.2. Why Scratch?

Scratch provides everything needed to begin developing computer games, multimedia presentations, interactive stories, graphic artwork, and computer animation.

Scratch can be used to play digital music and sound effects. Scratch's building block approach to programming sets it apart from other programming languages. This makes Scratch easier to learn. And yet Scratch provides plenty of programming power, allowing you to build very powerful application projects.

If you aspire to one day to become a professional programmer, you will find that Scratch provides everything needed to build a foundation from which you can make the transition. Scratch also packs all of the programming power and punch needed to satisfy the programming needs of most computer enthusiasts and hobbyists.

Scratch's slogan is *Imagine—Program—Share*! It is designed to encourage user's creativity by providing them with an easy to learn yet powerful programming environment in which they can unleash the power of their imagination. Scratch encourages and facilitates the development of application projects using a mixture of media, graphics, sound, and video in order to create something new.

1.2.3. Scratch's Building Block Based Approach

In text-based programming languages, code statements are formulated by following a complex set of syntax rules. Failure to precisely follow these rules when writing statements leads to errors that prevent applications from running. Scratch, on the other hand, uses a different approach. Scratch application projects are built by selecting and snapping together graphical programming blocks, as demonstrated in Figure 1.

By using code blocks in place of complex program text statements, Scratch significantly simplifies application development while still making use of the same basic programming logic and concepts implemented in other programming languages. As Figure 1 demonstrates, each code block represents a different command or action. Blocks fit together like pieces in a puzzle. You can only snap together blocks in ways that make syntactic sense, completely eliminating syntax errors that proliferate in other programming languages.

Some code blocks are configurable, allowing you to specify things like the number of times an action should execute, text that is to be displayed, or the color to be used when displaying something on the screen. Despite its use of graphical code blocks, Scratch supports the same basic set of programming techniques and constructs as do other traditional programming languages. For example, Scratch supports variables, conditional and iterative logic. Scratch also supports the manipulation of graphics and the integration of sound into application projects.

1.2.4. Installing Scratch

Before you can use Scratch, you need to install it on your computer. The installation process varies, depending on whether you use Microsoft Windows, Linux or Mac OS X. Instructions for installing Scratch on all these operating systems are provided. You can download a copy of Scratch from the Scratch website by executing the following steps:

1. Go to *http://scratch.mit.edu* and click on the *Download Scratch Now*! link.
2. The Download Scratch page appears. Fill in the optional form to receive updates about Scratch.
3. Click on the Continue to Scratch Download button. Click on the appropriate link for your operating system.

1.2.5. Creating a New Scratch Project

All new Scratch projects automatically contain a single sprite, representing an image of a kitten. By default, the sprite, named Sprite1, does not have any scripts but does have two costumes and two sounds associated with it. Using this sprite, let's create a Scratch application project that makes the kitten meow and say ''Hello World!'' when clicked.

The first step in creating a new Scratch application is to click on the *New* button located at the top of the Scratch IDE. In response, Scratch will create a new project, as shown in Figure 2. As Figure 2 shows, the Scratch IDE is organized into a number of separate components. For starters, the code block area contains code blocks, organized into ten different collections. You will use selected code blocks to create a script that makes the kitten talk.

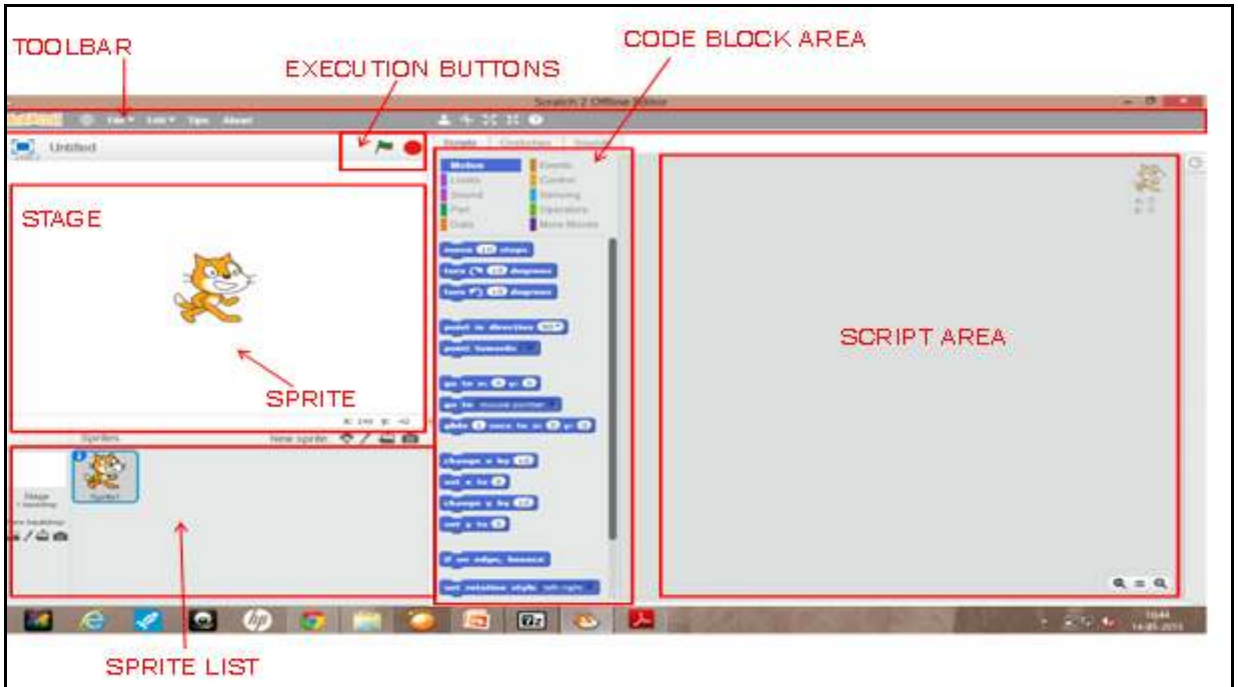


Figure 2 : The Scratch Interface

1.2.6. Changing Sprite Attributes

The application project that you are creating is designed to work with the default sprite. Rather than use the sprite's default name of Sprite1, let's assign it a more descriptive name. To do so, overtype the text displayed at the top of the sprite area with the word Cat. Once you change the name assigned to the sprite, the name change will automatically be reflected in the sprite list. If you look at the entry for the sprite in the sprite list, you should see a picture of the sprite, its new name, and the number of costumes currently assigned to the sprite (you can click on the Costumes tab at the top of the sprite area to view the sprite's costumes) as shown in Figure 3.

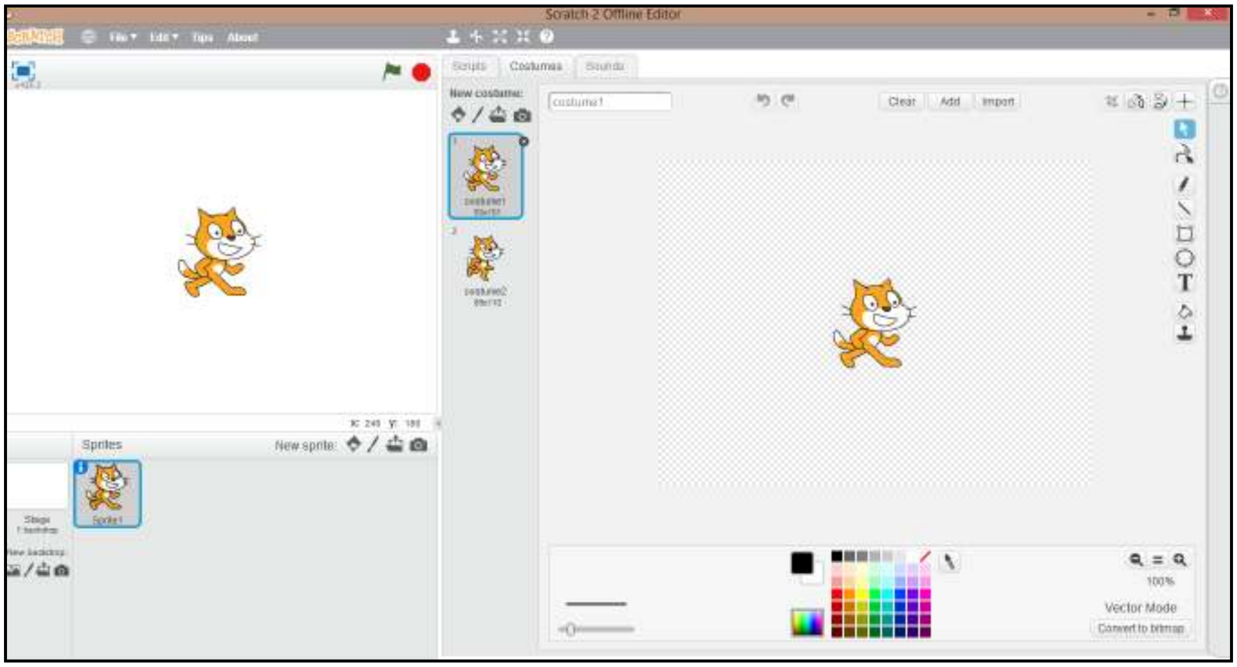


Figure 3: The *Costumes* Option in Scratch

1.2.7. Adding Code Blocks

1.2.7.1. Motion

Motion blocks control a sprite's placement on the stage. Motion blocks are coloured blue. There are motion blocks that let you set the direction a sprite will move and then other blocks to move them. There are also motions blocks that report on a sprite's location and direction.

Now that you have changed the name of the sprite, it is time to add the code blocks required to make the cat move 10 steps. Let's begin by clicking on the Motion button located at the top of the code block area. This displays a collection of code blocks that control the motion. Locate the code block labeled move 100 steps and drag and drop it onto the sprite area, as shown in Figure 4. Now when you click that code block, it can be seen that the sprite moves 10 steps.

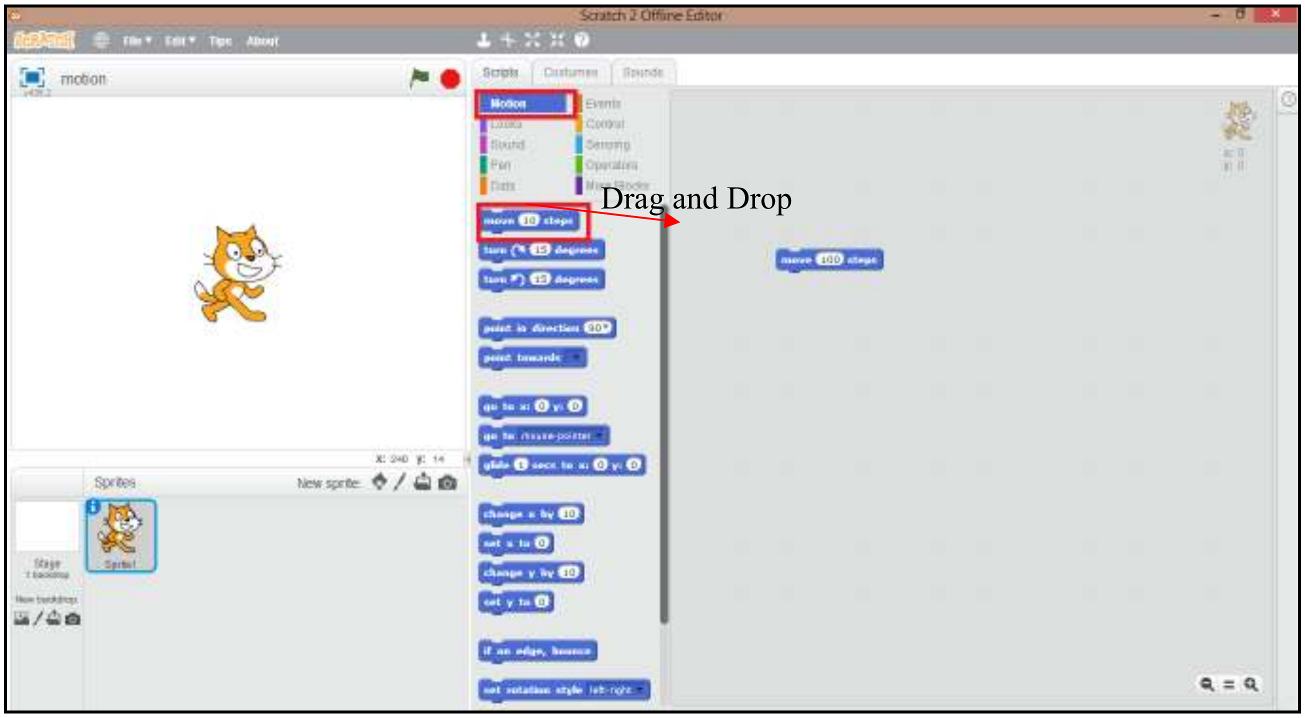


Figure 4: How to Use the *Motion* Code Block

1.2.7.2. Looks

Looks blocks modify sprite and background appearance and display text within popup bubbles. Looks blocks are colored purple. There are looks blocks that let you modify sprite costumes and colors. There are also blocks that let you modify a sprite's size and control whether a sprite is visible on the stage.

Let us now see a sample code to make the sprite say 'Let's go!' for 2 seconds. For that, create the code as shown in Figure 5. Now when you press the green flag, in the execution area, you can see the sprite saying, 'Let's go!'.

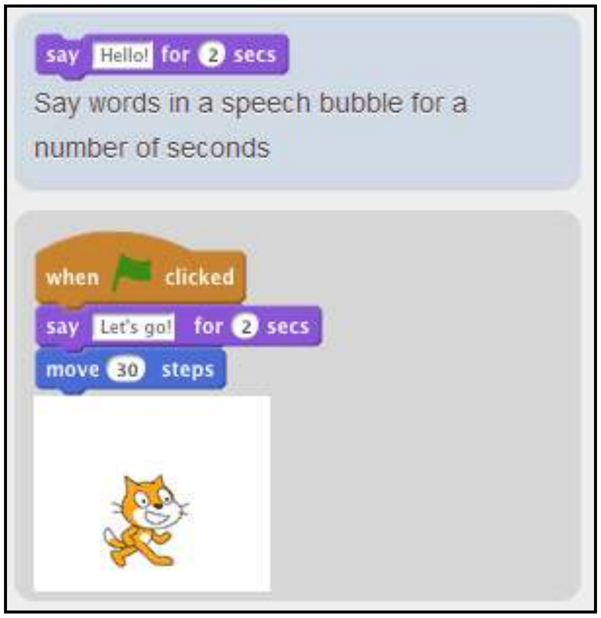
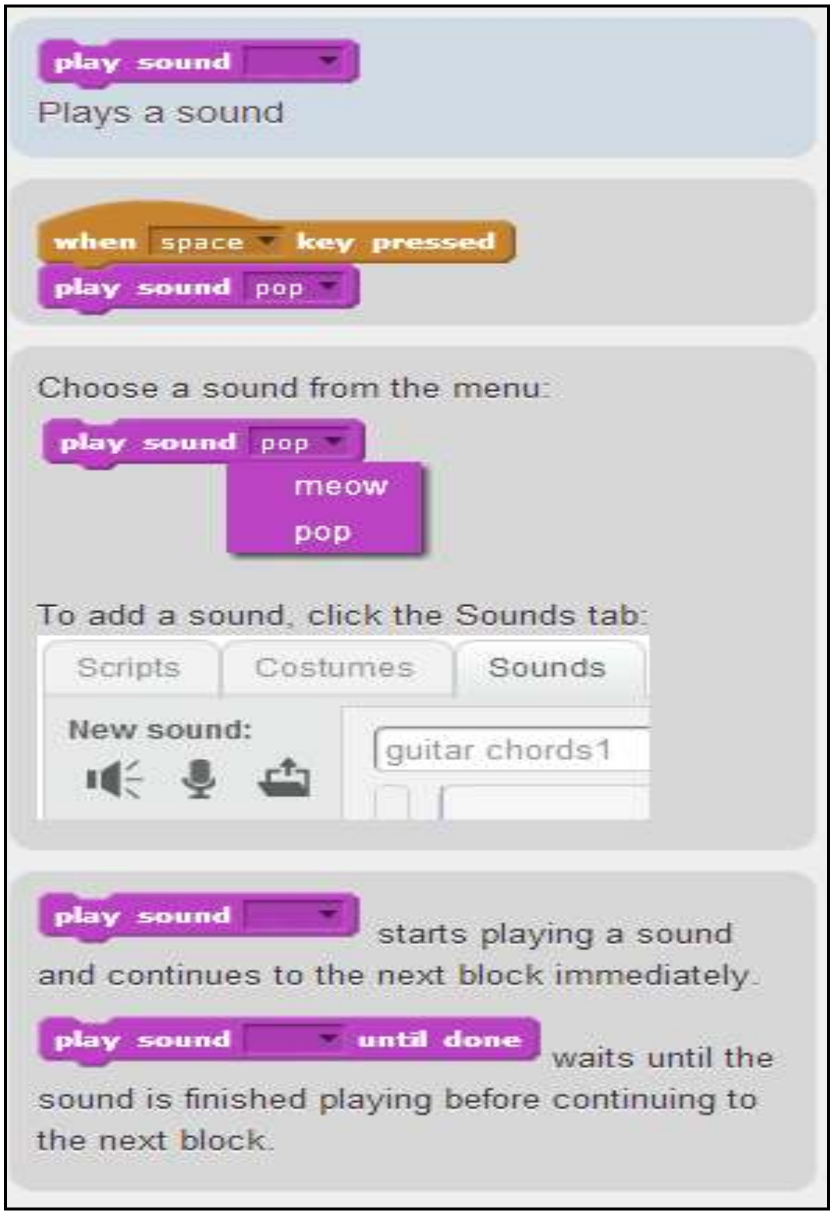


Figure 5: An Example on *Looks* Code Block

1.2.7.3. Making Some Noise

Sound blocks play music and add sound effects to your Scratch application projects. Sound blocks are colored pink. There are sound blocks that let you play sounds and drum beats, select different types of instruments, control playback volume, and modify tempo. Figure 6 gives an idea on how the sound block ‘play sound’ functions.



play sound

Plays a sound

when space key pressed
play sound pop

Choose a sound from the menu:

play sound pop

To add a sound, click the Sounds tab:

play sound starts playing a sound and continues to the next block immediately.

play sound until done waits until the sound is finished playing before continuing to the next block.

Figure 6: An Example of *Sound* Code Block

1.2.7.4. Drawing Lines and Shapes

Pen blocks draw any combination of shapes and lines using a virtual pen. Pen blocks are coloured mint green. There are pen blocks that let you enable and disable drawing, set colour and pen size, and apply shading. An example on how to draw a line using pen is illustrated in Figure 7.

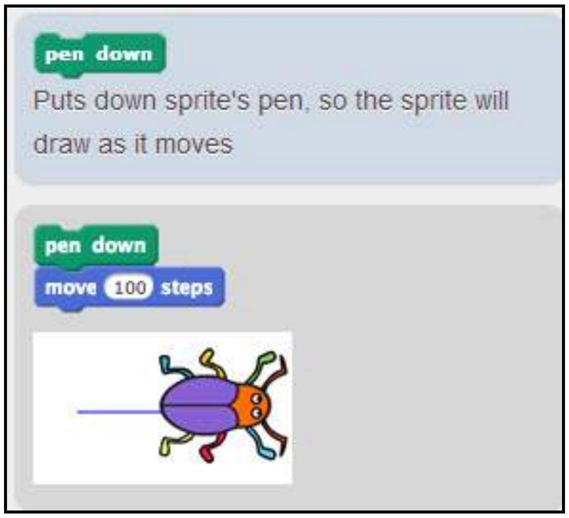


Figure 7: An Example on How to Use *Pen*

1.2.7.5. Looping, Conditional Logic and Event Programming

Control blocks automate the execution of scripts, pause script execution, and send messages to other sprites, allowing sprites to synchronize their execution. There is also control block that let you set up loops to repeatedly execute collections of code blocks as well as control blocks that let you conditionally execute other code blocks based on whether or not a test condition evaluates as true. Control blocks are colored gold.

The code snippet given in Figure 8 is an example of a simple loop that repeats for 10 times. The sprite moves 10 steps repeatedly for 10 times. Also Figure 9 is an example of implementing conditional statements, i.e. in creating blocks which check for certain conditions.

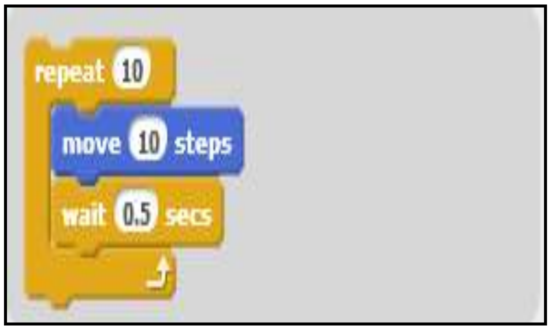


Figure 8: An Example on Repetition

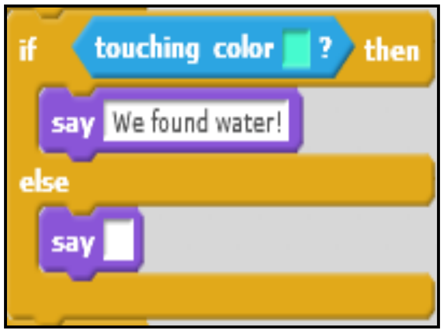
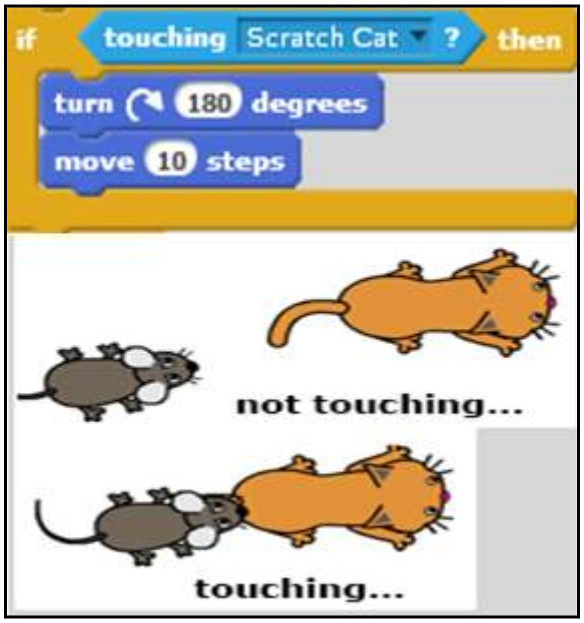


Figure 9: An Example of conditional statements

1.2.7.6. Sensing Sprite Location and Environmental Input

Sensing blocks determine the location of the mouse-pointer, its distance from other sprites, and whether a sprite is touching another sprite. A basic example of sensing is displayed in Figure 10. Here there are two sprites, 'Mouse' and 'Scratch Cat'. The code shown in Figure 10 is the script written for the sprite 'Mouse'. If the 'Mouse' sprite touches the 'Scratch Cat', then, the 'Mouse' sprite has to turn 180 degrees and move 10 steps.



```
if <touching [Scratch Cat ▾] ?> then
    turn ↻ (180) degrees
    move (10) steps
```

Figure 10: An Example for Sensing

1.2.7.7. Working With Numbers

Data and operators blocks perform arithmetic operations, generate random numbers, and compare numeric values to determine their relationship to one another. Data block is orange in colour and the operators block is in green. There are numbers blocks that can be used to round numeric values and to execute a host of mathematical functions like determining absolute value or square root of a number.

Data blocks store and retrieve numeric values in computer memory. You will need to use variables to store data as your application executes. For example, if you create a game that challenges the player to try and guess a randomly generated number, you will need to use a variable to store and refer back to this number.

Variables (a value that varies), can be used in conjunction with conditional programming logic to control the execution of other code blocks. Variables can also be used to control the repeated execution of code blocks embedded within code block loops. You can create and name custom variables blocks and assign them a starting value. You can also modify their values during script execution. Other code blocks can retrieve variable values and use them as input.

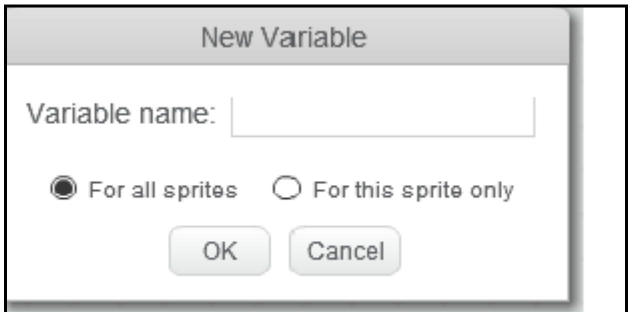


Figure 11: The Window for Giving Variable Name

Figure 12 shows a variable 'count' that has been created. For that, press the Make a Variable button, then a pop up window as in Figure 11 shows up, where the variable name can be specified.

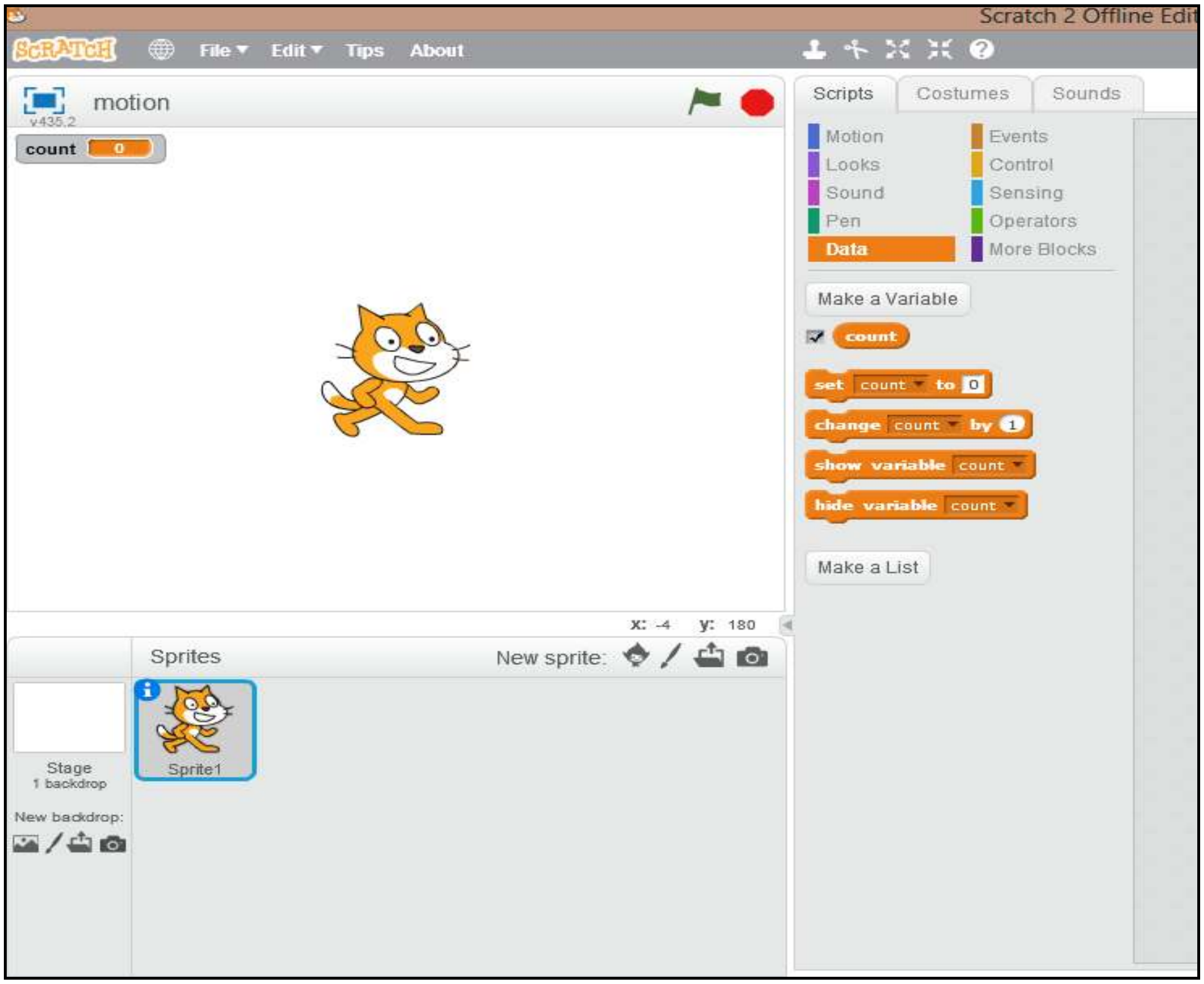


Figure 12: Creating the Variable Named 'count'

- **How to Create a Simple Addition Project**

Create 3 variables with the name, *a*, *b* and *sum*. Then set the value of *a* to 5 and *b* to 10. Finally, set the value of *sum* as the sum of *a* and *b*. The code is shown in the *script area* of Figure 13 and the result of the operation can be seen in the *stage area* of Figure 13.

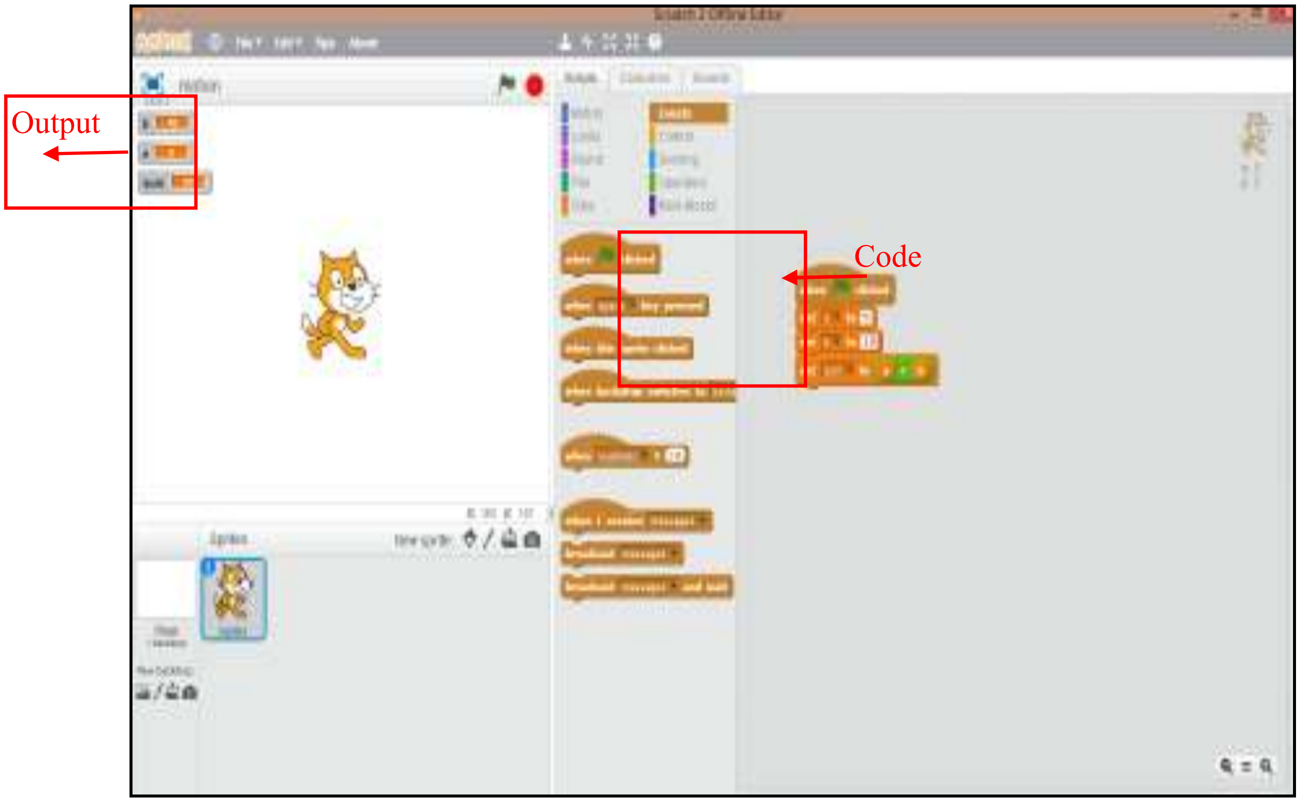


Figure 13: The Output for the Addition Project

1.2.7.8. Modularization

In Scratch, it is possible to create and define our own modules and use those modules whenever required. It is provided in the 'More Blocks' option given in dark violet colour as shown in Figure 14.

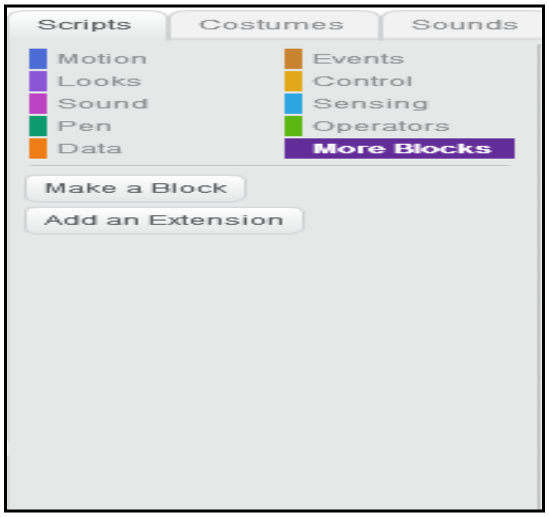


Figure 14: The 'More *Blocks*' Code Block

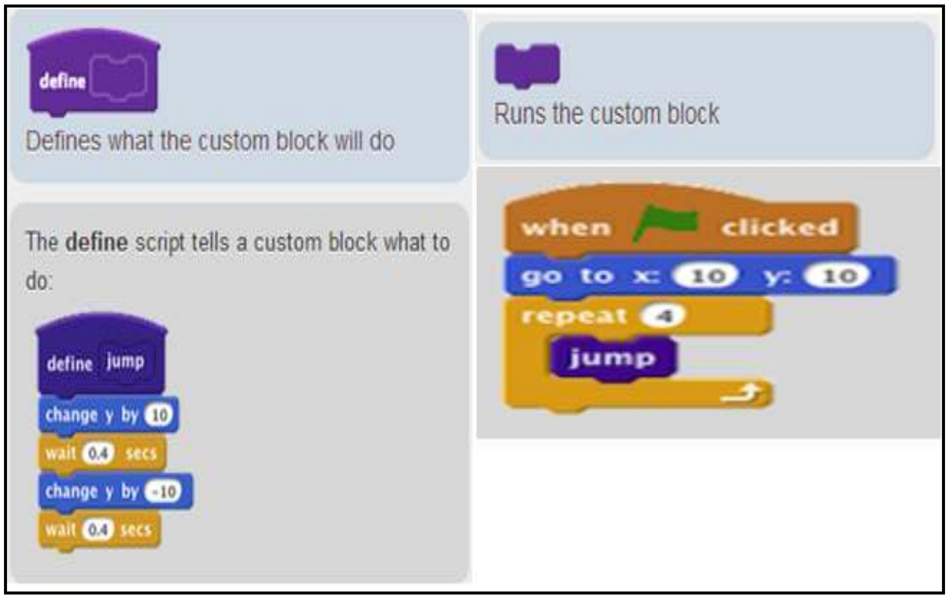


Figure 15: An Example for the Define Code Block

A simple example on how to use this facility is shown in Figure 15 where a '*jump*' module has been created and defined under the '*define*' function and later used in the code.

1.2.8. The Scratch Website

The most informative and helpful Scratch website is the Scratch site developed and maintained by MIT located at http://scratch.mit.edu. This site is packed with helpful information, including documentation, video tutorials, and forums where you can go to interact with and learn from other Scratch programmers from around the world. Best of all, this site provides instant access to tons of Scratch projects, all of which you can download, experiment with, and learn from.

1.3. Raptor

1.3.1. Introduction

Raptor is a simple to use tool to generate executable flowcharts. It helps to develop problem solving and algorithmic thinking skills by easy visualization of the execution of a logic structure. It was originally developed by and for the Department of Computer Science at the US Air Force Academy, but is now used for CS education in over 28 countries on at least 4 continents and is freely distributed as a service to the CS education community.

1.3.2. Installation

For installing Raptor in your system, you will need to follow the following steps:

1. You can see links to download the latest Raptor package from the homepage http://raptor.martincarlisle.com
2. You can download either a Windows version (preferred) from http://raptor.martincarlisle.com/raptor_2014.msi or a Portable version from http://raptor.martincarlisle.com/RaptorPortable_4.0_Revision_6.paf.exe
3. If you going to use the Windows version of Raptor, you will be required to have .NET 3.5+ installed for Windows 8.
4. If you are going to use Raptor on a Linux machine, download the zip file from http://raptor.martincarlisle.com/RAPTOR_exe.zip, extract it, and run the Raptor.exe file inside the extracted folder using Mono Runtime. The set of functionalities available for the Linux version of a raptor is somewhat limited to the Windows version.

1.3.3. Interface Overview

The user interface of Raptor will look as shown in Figure 1 and Figure 2 below. It has two windows – a main Raptor window and a Master Console.

The main Raptor interface is divided into five regions namely Menu, Toolbar, Symbols, Watch Window and Workspace.

1. **Menu**
   The contents of the Raptor menu and their corresponding functions/options are as follows:

   1) File – Create a new file, open an existing file, print facilities etc.
   2) Edit – Comment, Cut, Copy, Paste, Delete, Undo, Redo etc.
   3) Scale – Different zoom options for the display.
   4) View – Different view options for the interface.
   5) Run – Run modes like Run step by step, Reset etc.
   6) Mode – To choose among three modes available – Novice, Intermediate and Object Oriented.
   7) Ink – To choose what color ink you want to draw around the flowcharts – just like MS Paint.

8) Window – Different window arrangement modes.

9) Generate – To convert Raptor flowchart to computer programs in a set of computer languages like Ada, C#, C++, Java and as a standalone executable application.

10) Help – An inbuilt self-help manual.

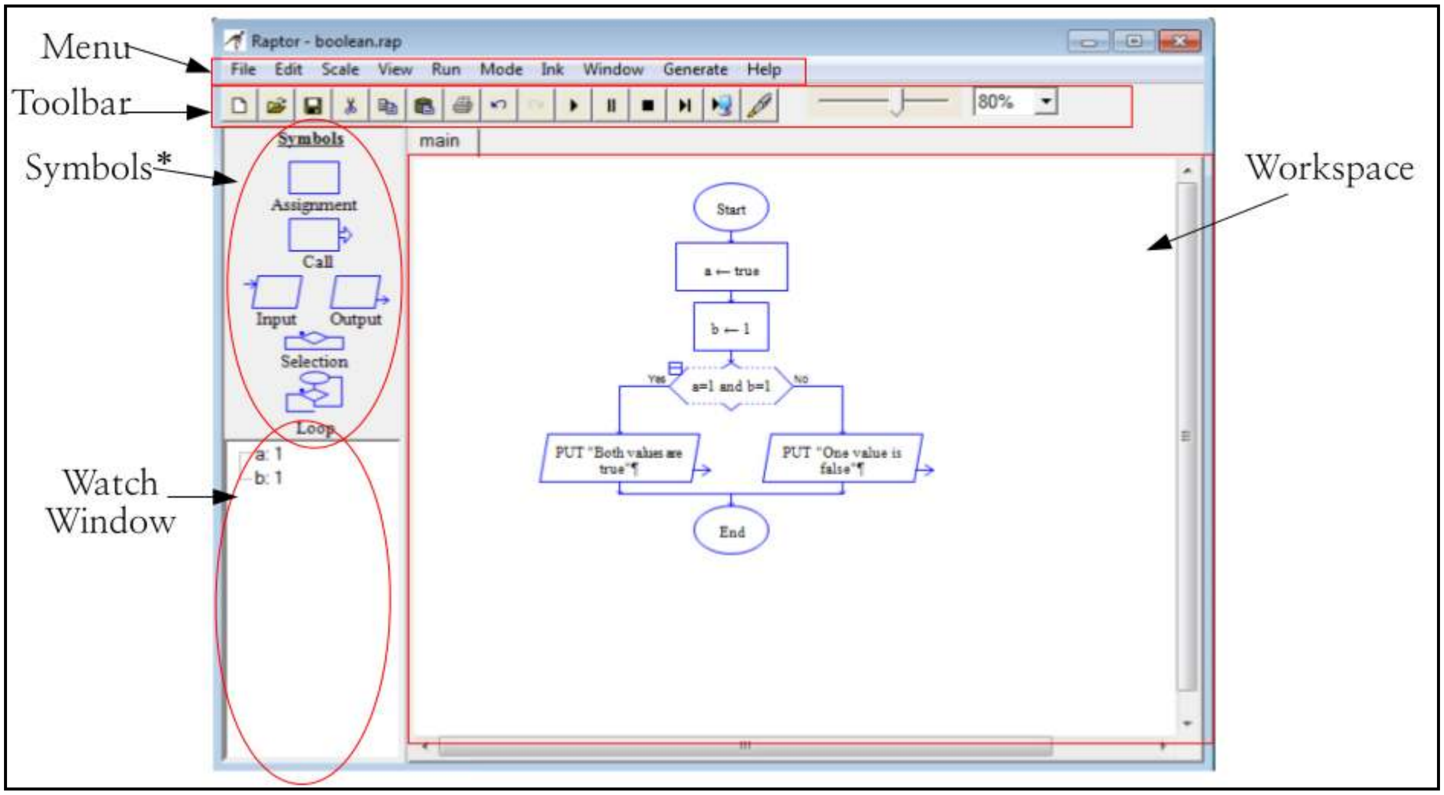


Figure 1: Main Raptor window

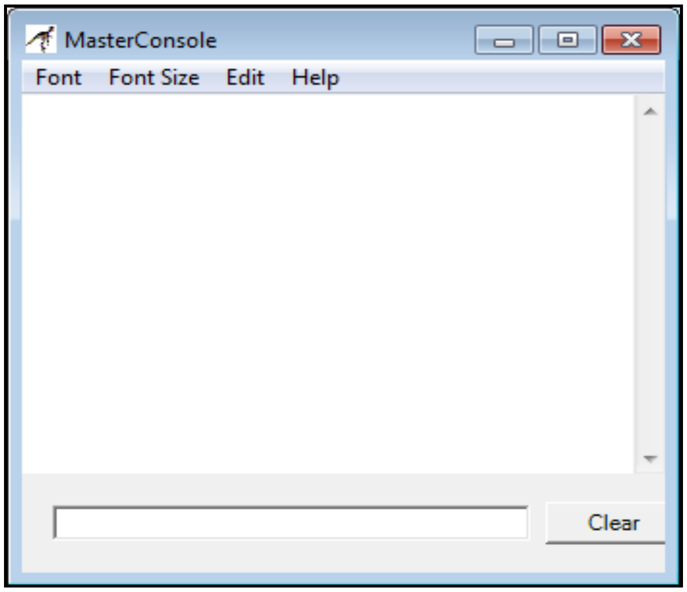


Figure 2: Master Console

2. **Toolbar**
   The toolbar in Raptor has a set of frequently accessed menu items represented using icons.
3. **Symbols**
   Symbols section given at the left of the main Raptor window displays a set of logical building blocks using which you can build the flowchart. The set of symbols made available in Raptor are mentioned below along with their purpose.

   1) Assignment – To assign a value to a variable.
   2) Call – To call a function.
   3) Input – To receive an input from the user while execution.
   4) Output – To output something to the user typically via the Master Console.
   5) Selection – To implement a logical condition.
   6) Loop – To implement repeated sets of logic statements.
4. **Watch Window**
   Watch window displays the updated value stored in all variables during execution.
5. **Workspace**
   This is where you build the flowchart using the elements from the Symbols section.

1.3.4. Simple flowchart building

Let's take an example (**Example 1**) of building a flowchart which takes two numbers as input to a flowchart and gives their sum as the output. Here's how we do it step by step.

1. Open Raptor. We can see both the main Raptor window and Master Console come up. As seen in Figure 3, the watch window will have the "Start" and "End" blocks by default.

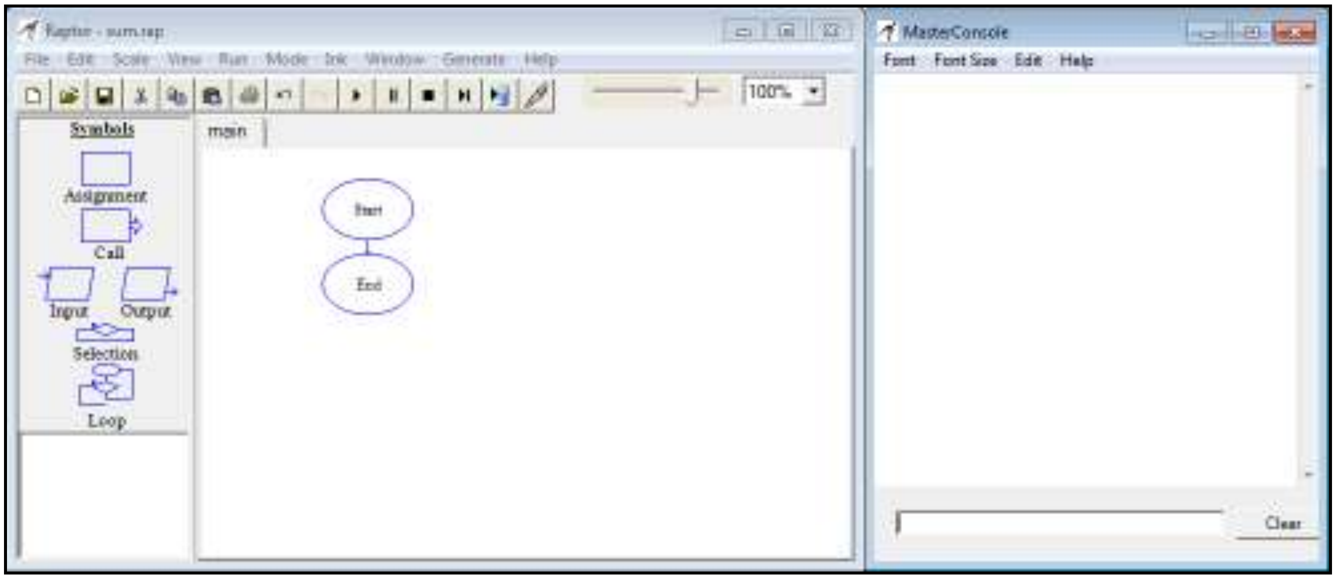


Figure 3: Raptor Window and Master Console

2. We need to get the values of the two variables entered into the flow chart. So we need two “Input” blocks added to the flowchart. Click on the “Input” block on the “Symbols” section and it would turn red as in Figure 4 below. The part is shown rounded in a dark circle.

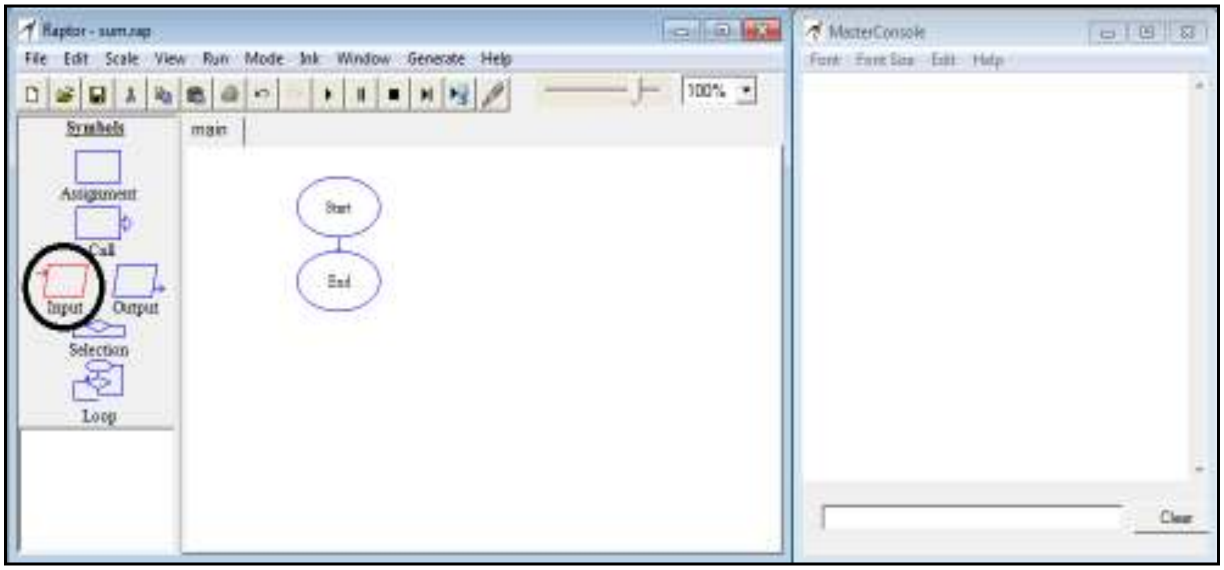


Figure 4: Selecting an “Input” block

3. Once the “Input” block turns red, click on the part of the flowchart where we want the block to be placed. In our case, it will be on the “arrow” connecting the “Start” and “End” blocks. Once clicked, the block will be placed there as shown in Figure 5 below.

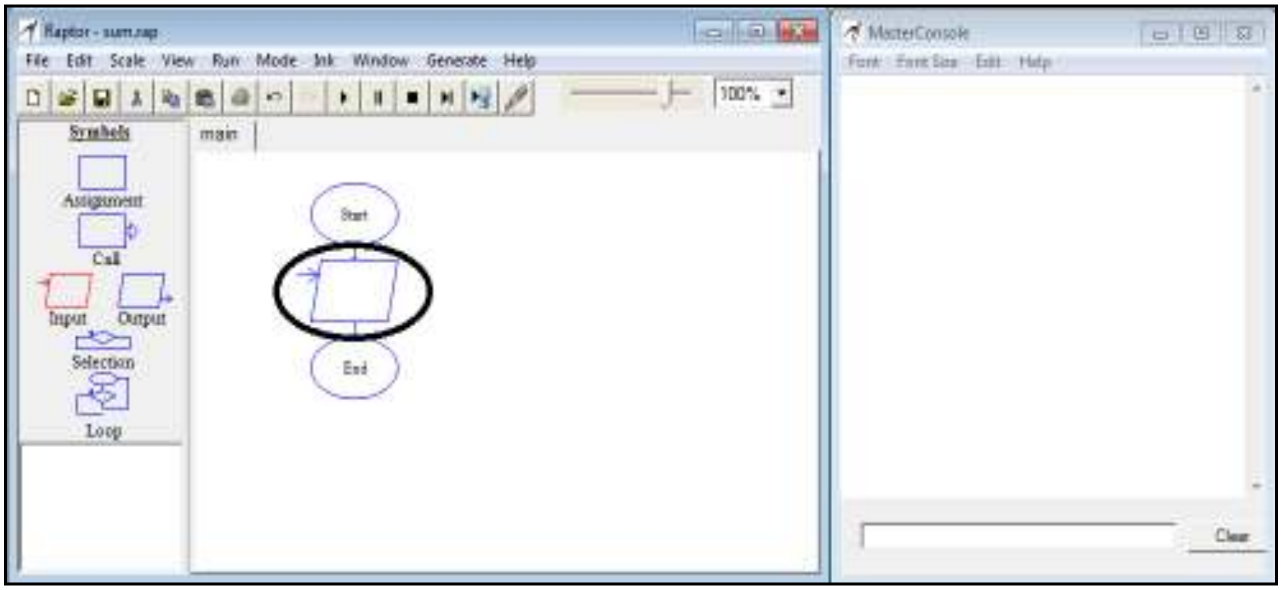


Figure 5: “Input” block placed in the flowchart

4. Now, we need to configure the “Input” block to receive an input and to store it in a variable. To do that, we should double click on the block and it would turn red and opens a menu as shown in Figure 6.

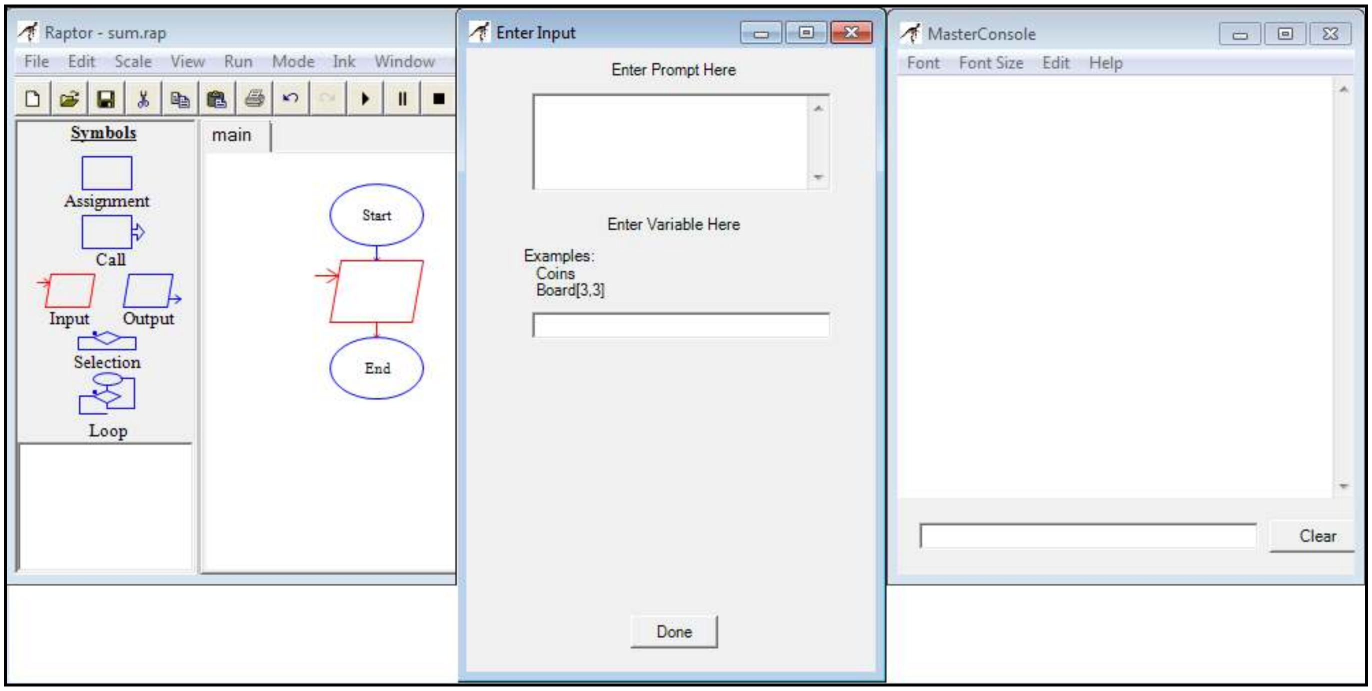
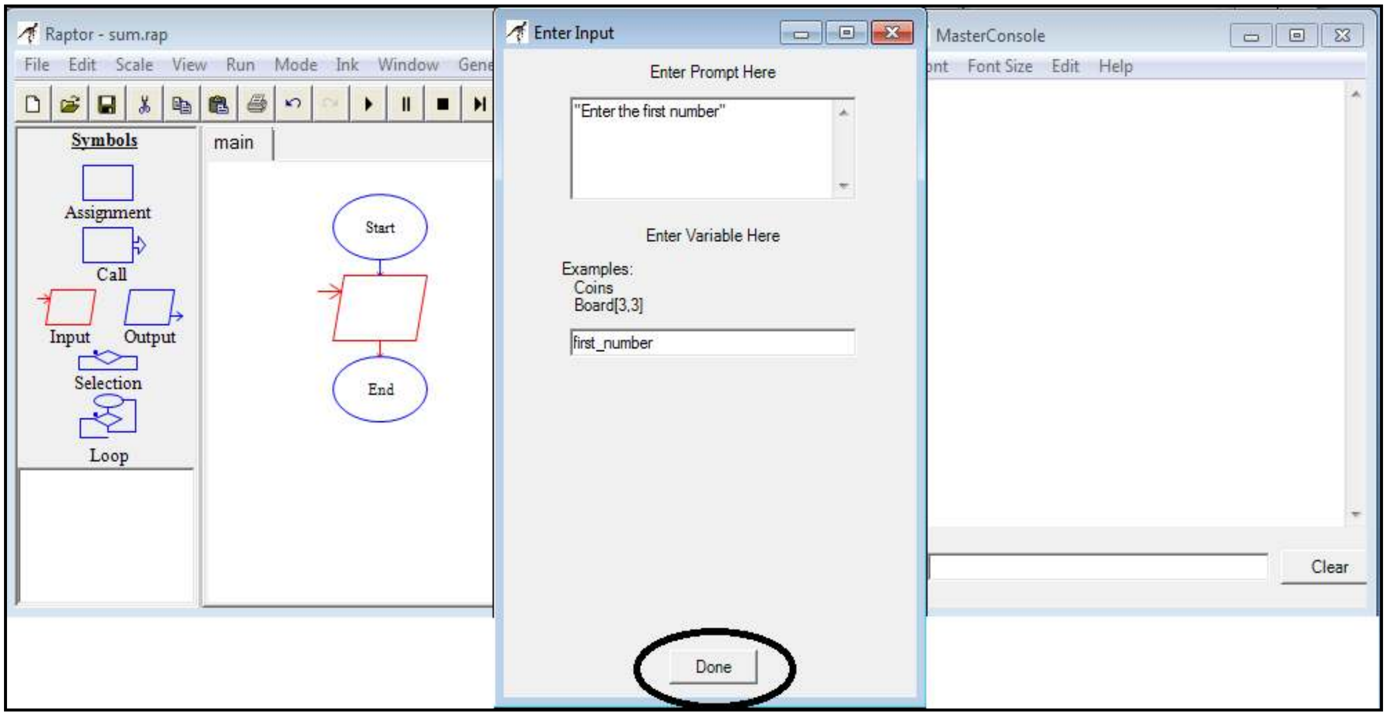


Figure 6: "Enter Input" menu

5. In the "Enter Prompt Here" text box, we should enter the message (within double quotes) that we want to print to the user while asking for the input. In the "Enter Variable Here" text box, we enter the variable name where we want to store the value read. At last, press the "Done" button marked in the dark circle as shown in Figure 7.



"Enter the first number"

first_number

Figure 7: Finishing the configuration part

6. Now, we do the same procedure (steps 2 to 5) for reading the second number. After that is done, the Raptor window would look like Figure 8.

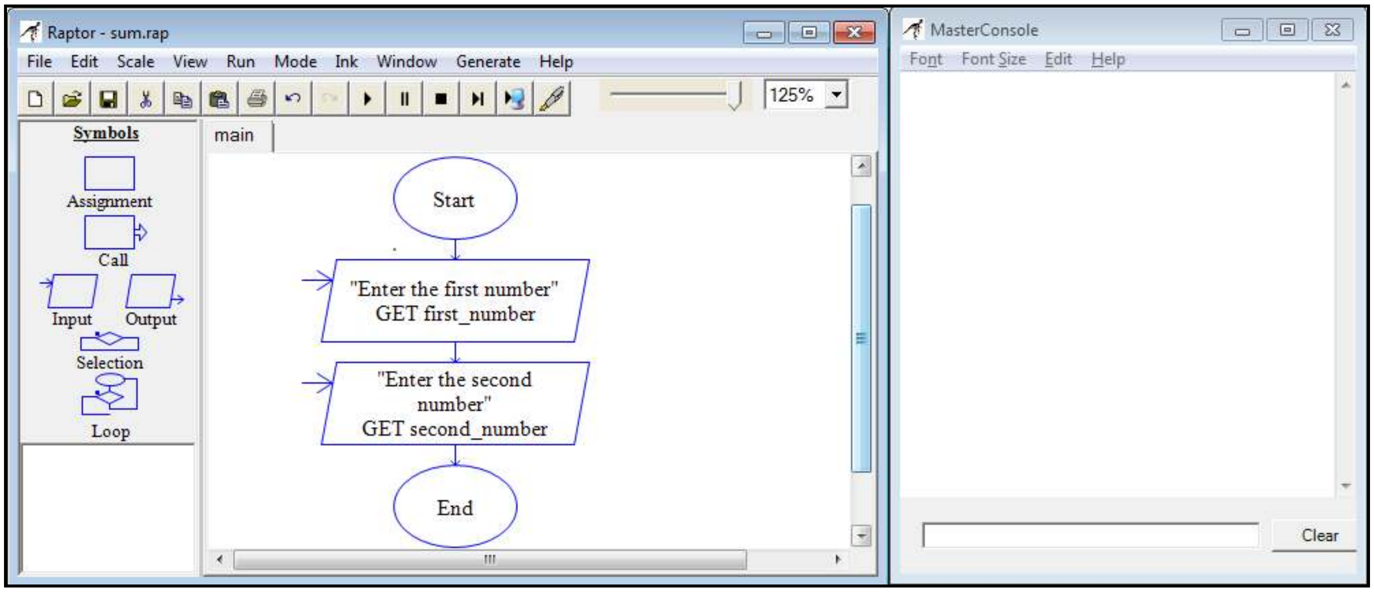


Figure 8: After entering both numbers

7. Now we need to do the operation of adding the values in the two variables and storing their sum in a third variable, say "third_number". We use the "Assignment" block from the "Symbols" section for this. As before, we click on the block and then click on the position where we need it to be placed in the flowchart – which is right under the second input block. The window looks like Figure 9 once this is done.

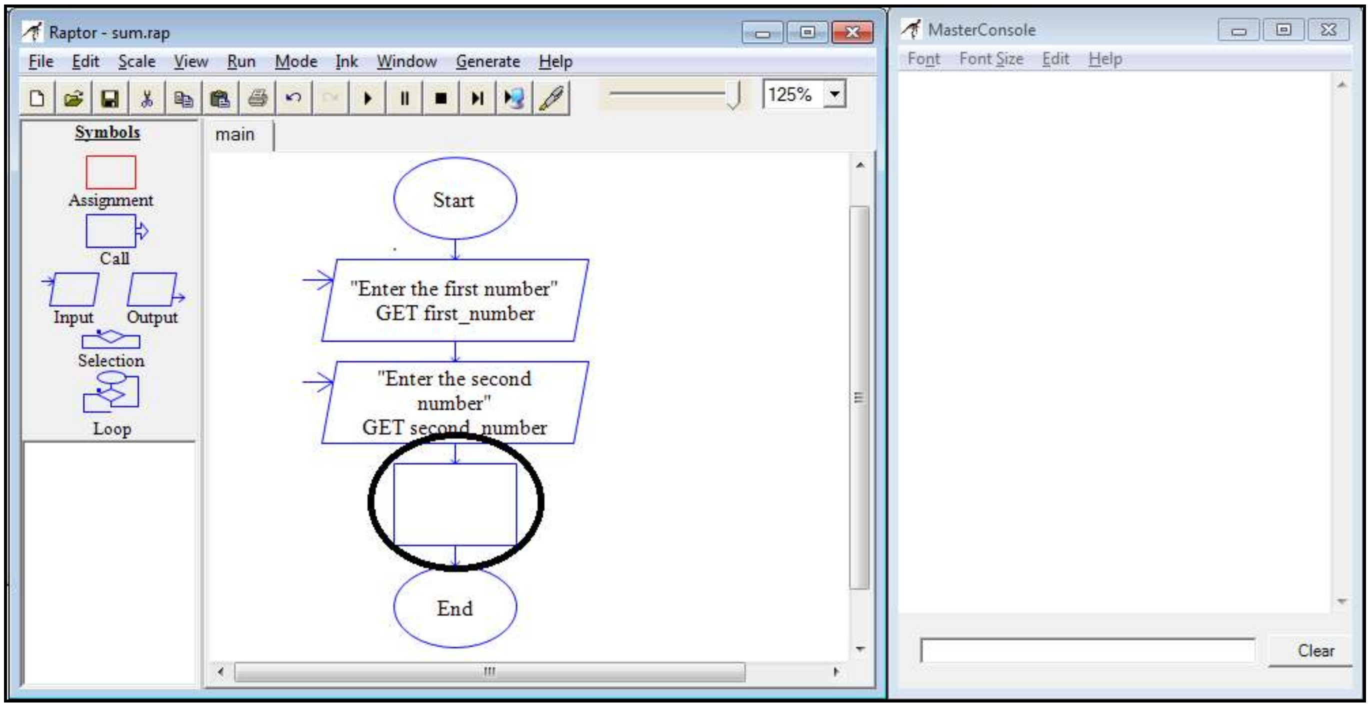
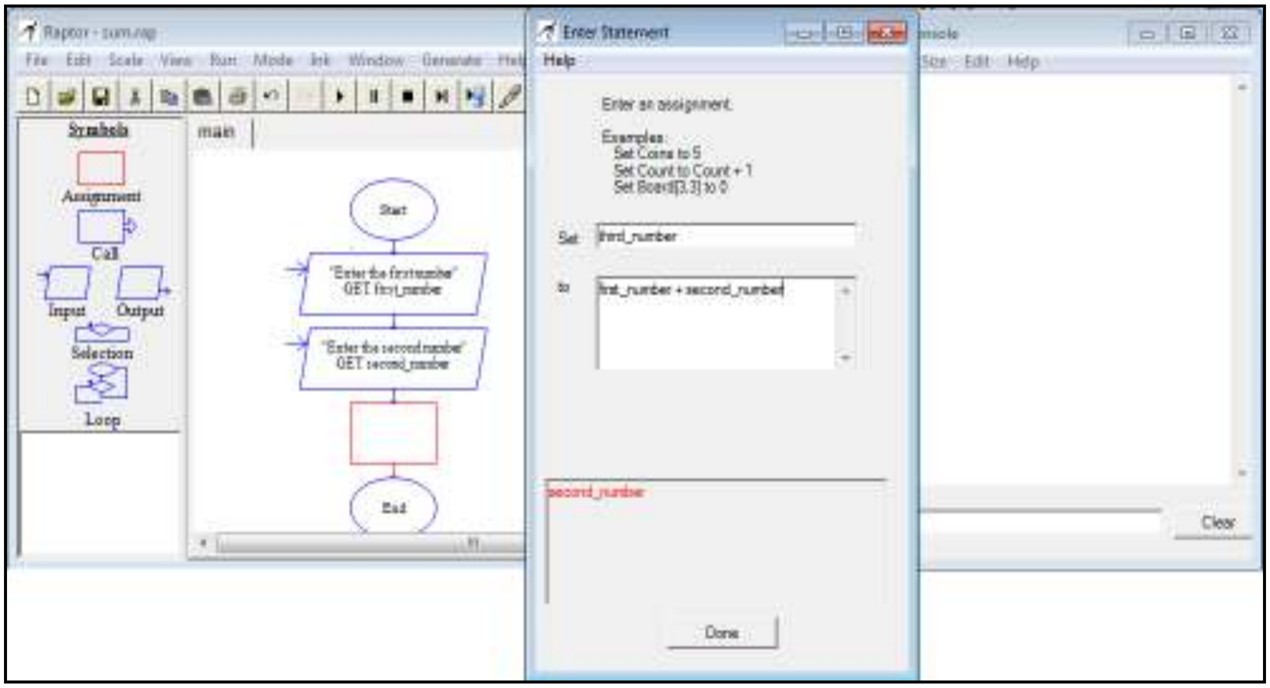


Figure 9: Adding "Assignment" block to the flowchart

8. We double click on the "Assignment" block to configure it to do the operations of finding the sum of the two numbers and assigning it to the third variable. Please refer to Figure 10 to see how this is done. After that click on the "Done" button.



Enter an assignment.

Examples:
Set Coins to 5
Set Count to Count + 1
Set Board[3,3] to 0

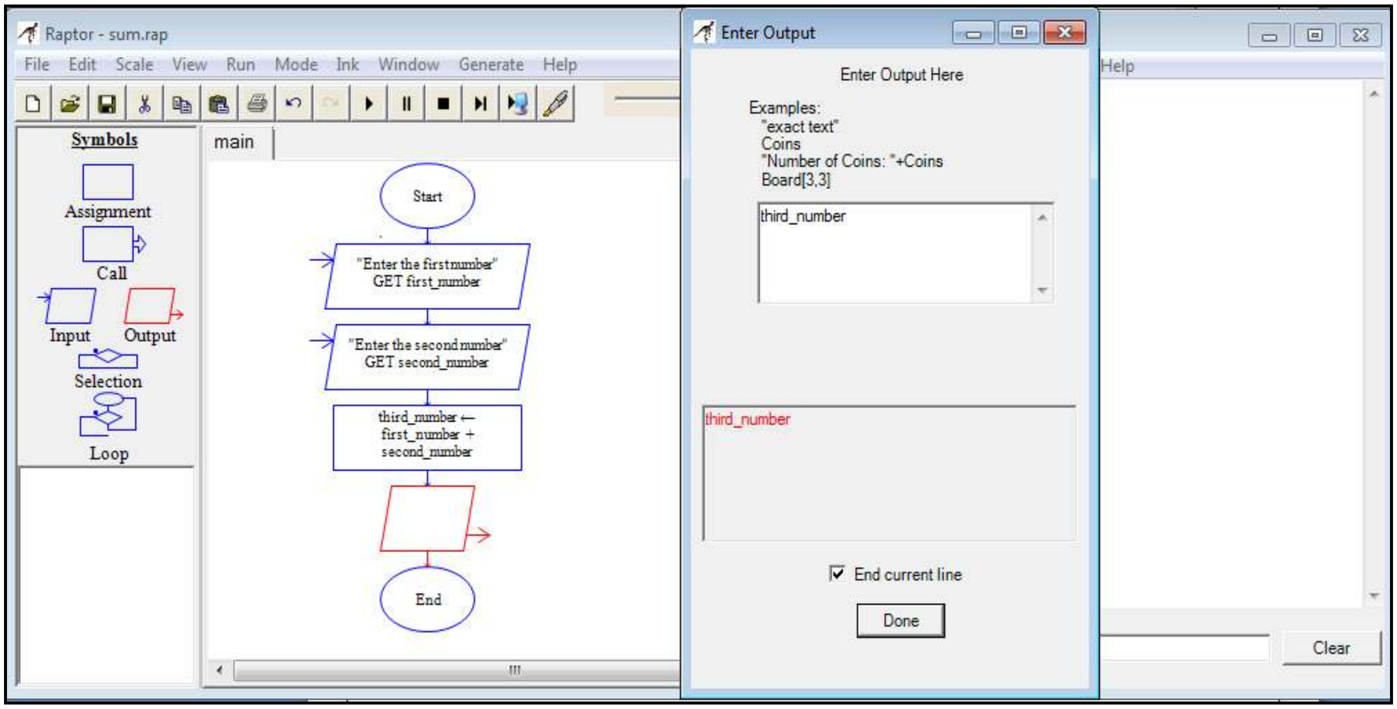
Set: third_number

to: first_number + second_number

second_number

Figure 10: Configuring "Assignment" block

9. Now we have to add the "Output" block to print the value stored in the third_variable. So we add the "Output" block and configure it to print the value stored in "third_number", as shown in Figure 11 below and click the "Done" button.



Enter Output Here

Examples:
"exact text"
Coins
"Number of Coins: "+Coins
Board[3,3]

third_number

third_number

[✓] End current line

Figure 11: Configuring "Output" block

10. Now go ahead and press the "Execute to Completion" button in the toolbar. It is shown inside a dark circle in Figure 12 below.

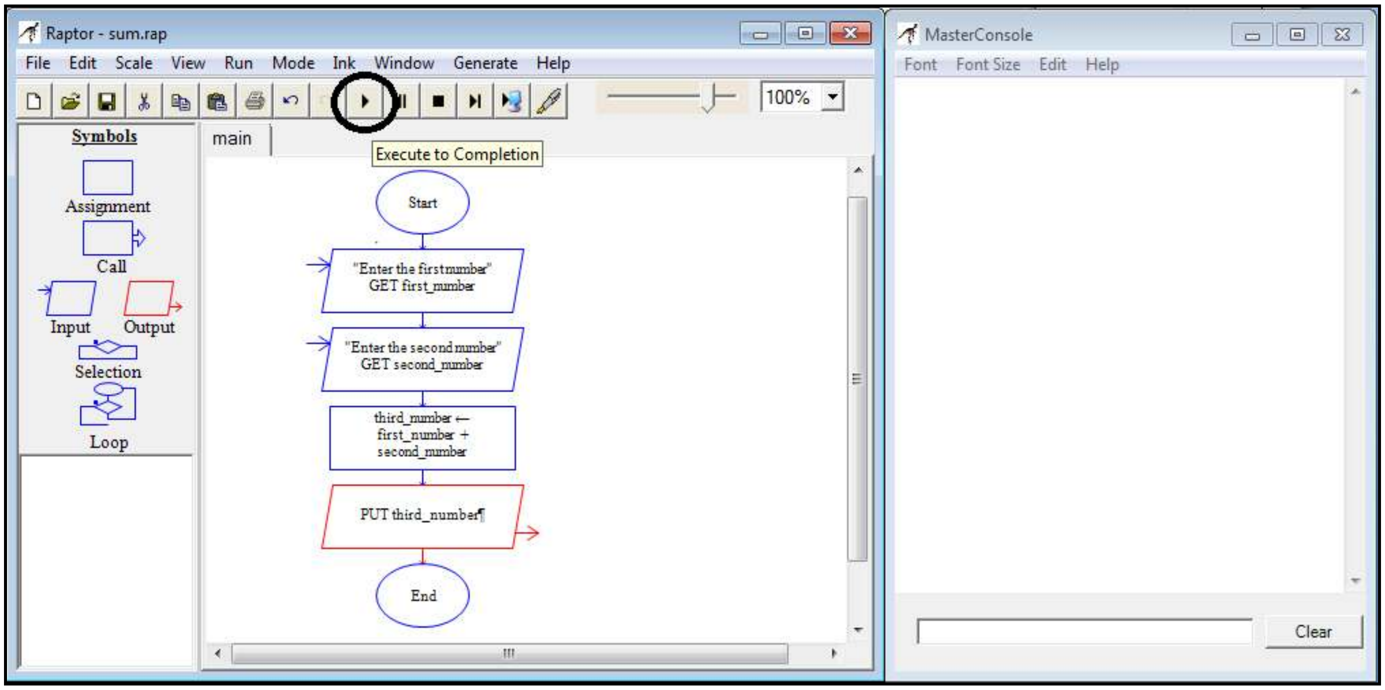

Raptor - sum.rap
File Edit Scale View Run Mode Ink Window Generate Help
100%
Execute to Completion
Symbols
Assignment
Call
Input
Output
Selection
Loop
main
Start
"Enter the first number" GET first_number
"Enter the second number" GET second_number
third_number ← first_number + second_number
PUT third_number¶
End
MasterConsole
Font Font Size Edit Help
Clear

Figure 12: The "Execute to Completion" button is shown encircled in the toolbar

11. Once the flowchart executing, the particular block under execution would be highlighted with a green color outline as shown in Figure 13. Once the input blocks are reached, two input window would pop up one by one asking you to "Enter the first number" and "Enter the second number" respectively. Fill in the text boxes with their corresponding numbers and press the "OK" button.

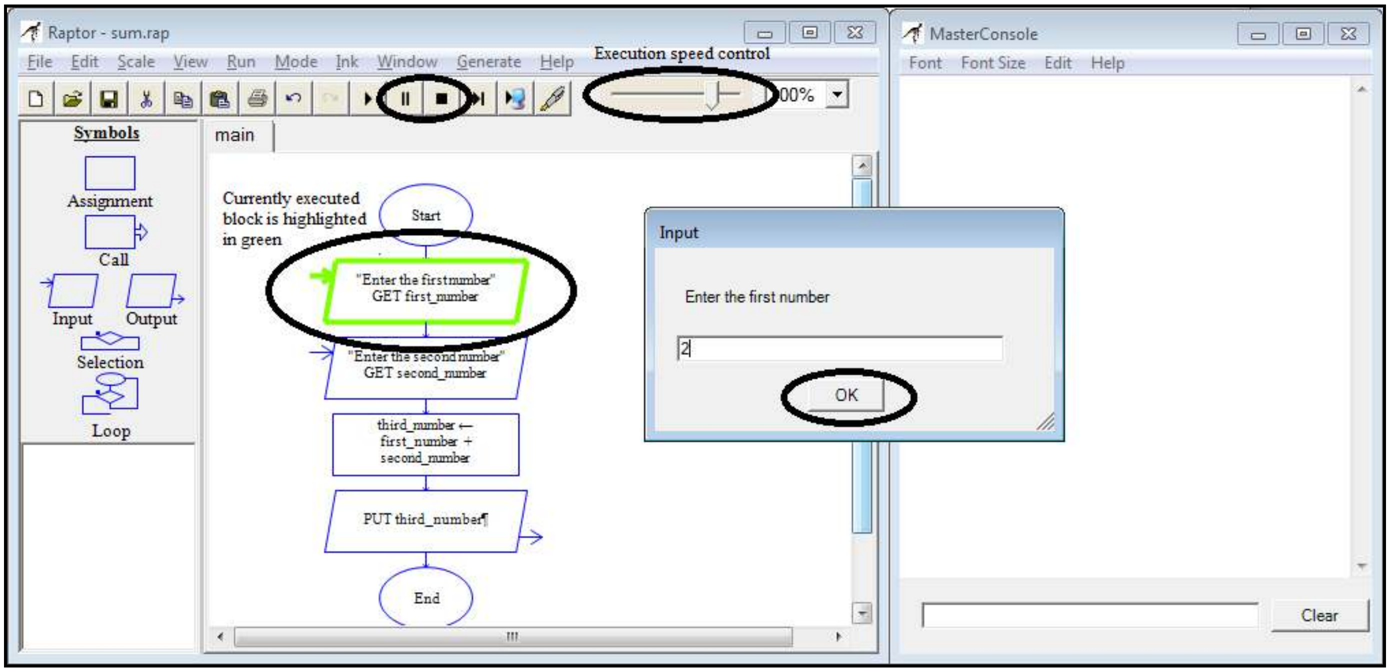
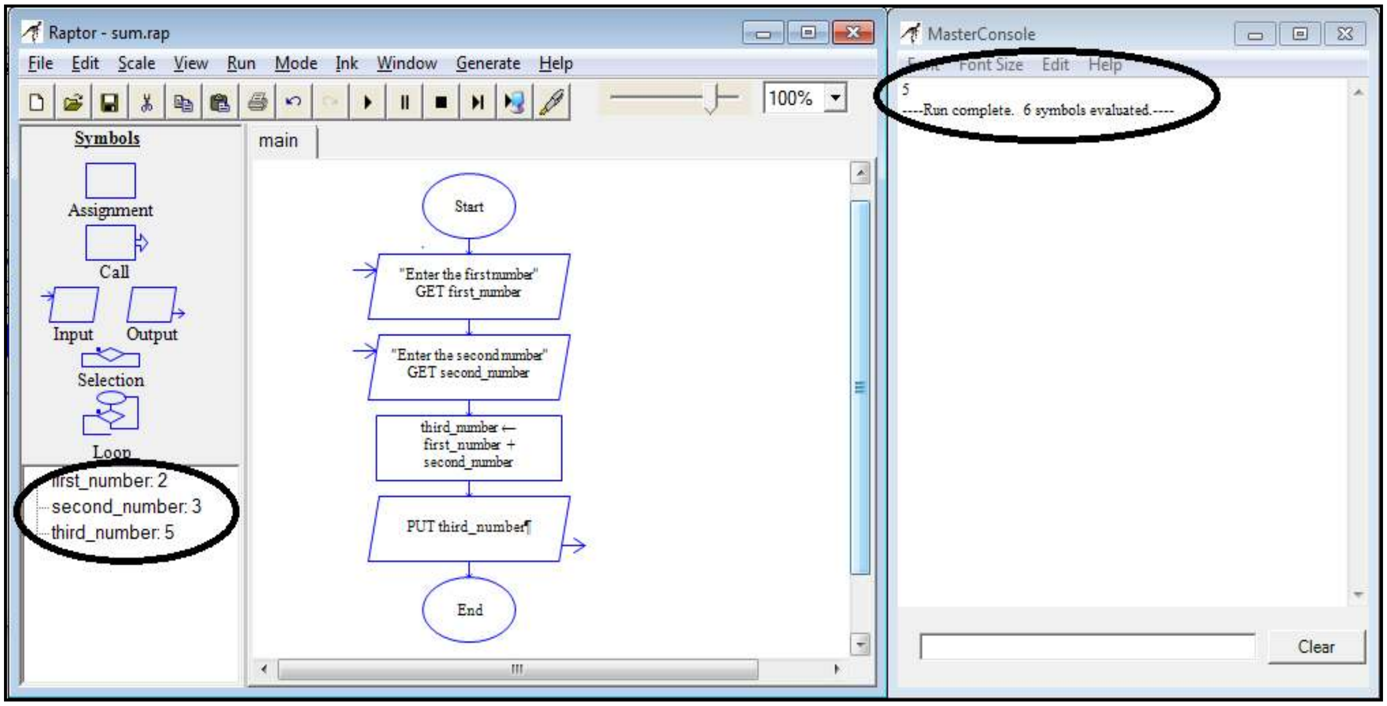

Raptor - sum.rap
File Edit Scale View Run Mode Ink Window Generate Help
Execution speed control
00%
Symbols
Assignment
Call
Input
Output
Selection
Loop
main
Currently executed block is highlighted in green
Start
"Enter the first number" GET first_number
"Enter the second number" GET second_number
third_number ← first_number + second_number
PUT third_number¶
End
Input
Enter the first number
2
OK
MasterConsole
Font Font Size Edit Help
Clear

Figure 13: The execution process

Please note that we can use the “Pause”, “Stop/Reset” buttons and the slider in the toolbar to control the algorithm execution process and the speed of execution respectively.

12. Once execution is completed, the result is displayed in the Master Console along with the status of execution, as shown in Figure 14.


Raptor - sum.rap
File Edit Scale View Run Mode Ink Window Generate Help
100%
Symbols
Assignment
Call
Input
Output
Selection
Loop
main
Start
"Enter the firstnumber" GET first_number
"Enter the second number" GET second_number
third_number ← first_number + second_number
PUT third_number¶
End
MasterConsole
Font Font Size Edit Help
Clear

```
first_number: 2
second_number: 3
third_number: 5
```

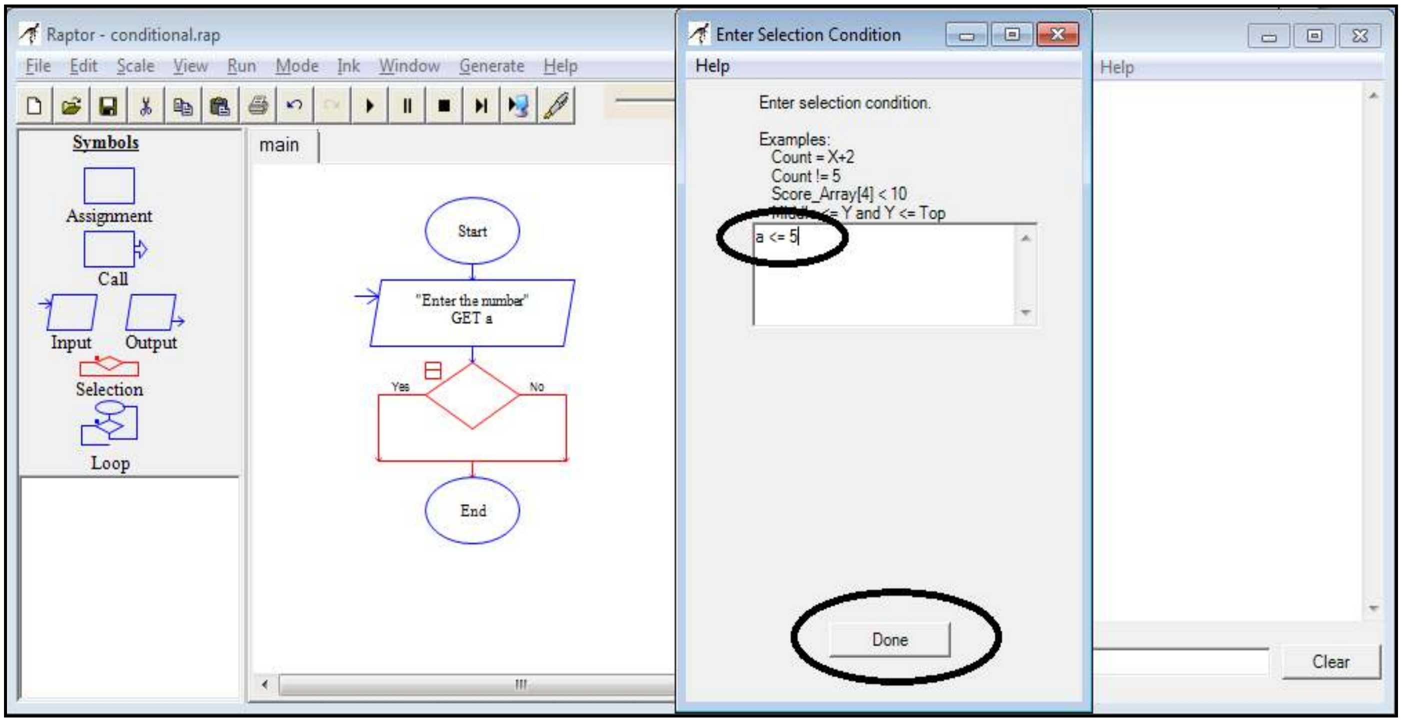
```
5
----Run complete.  6 symbols evaluated.----
```

Figure 14: The result displayed in the Master Console

It can also be noted that the “Watch Window” on the left now will be showing the latest values of all the variables used during the execution process. This “Watch Window” will in fact be updated at real time throughout the execution process.

1.3.5. Conditional Logic

To show how implementing Conditional Logic is implemented in Raptor, let’s take an example **(Example 2)** in which a value is read from the user and stored into a variable, say “a” and we need the flowchart to add 1 to “a” if it less than or equal to 5, and subtract 1 from “a” if otherwise. To implement conditional logic, the “Selection” block in the “Symbols” section is to be added to the flowchart. The procedure involved is same as steps 2 and 3 in the above explained example. After adding the “Selection” block, we need to configure it by double clicking on it and entering the condition check that we want to add. Figure 15 shows how to configure the “Selection” block.



Enter selection condition.

Examples:
  Count = X+2
  Count != 5
  Score_Array[4] < 10
  Middle <= Y and Y <= Top

a <= 5

Figure 15: Configuring the Selection block

Once the condition is added in the window and the "Done" button is pressed, we need to add logic to the "Yes" and "No" wings of the block in the flowchart. Once this is done, the flowchart would look like Figure.16.

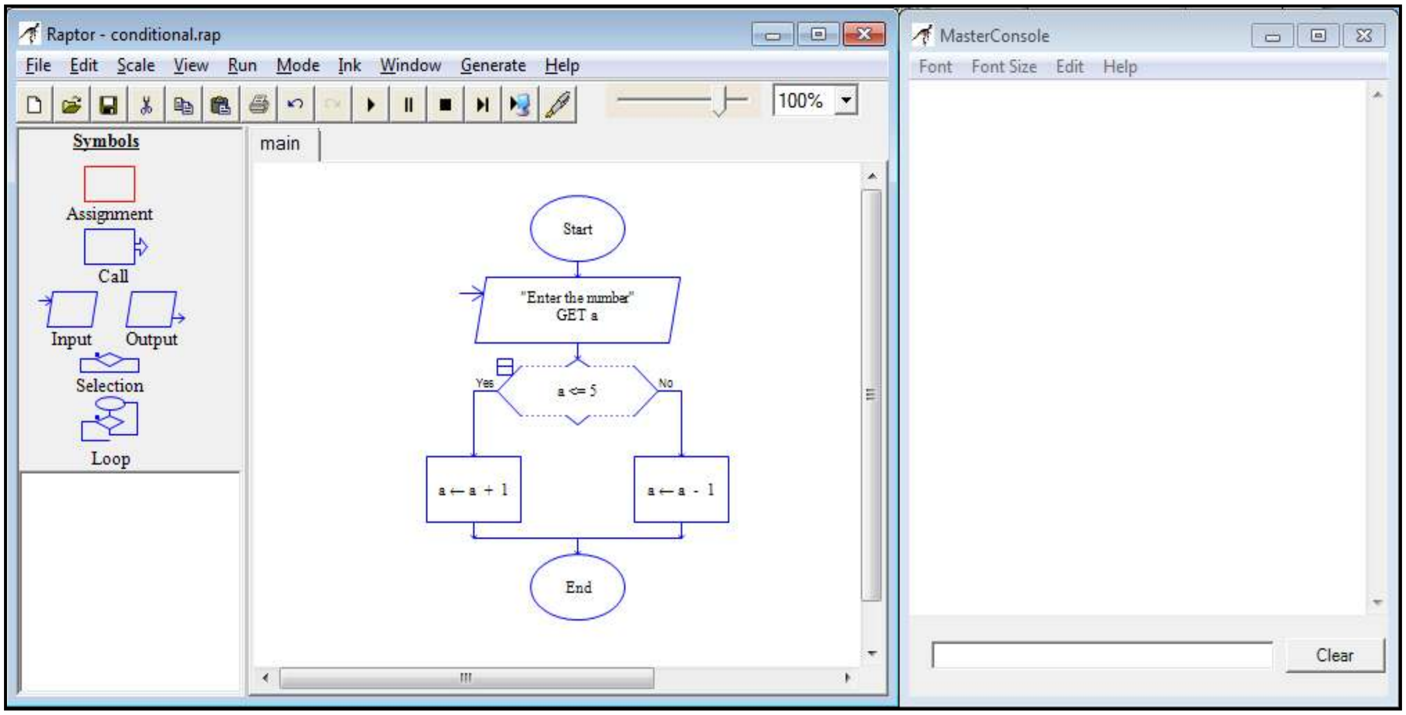


Figure 16: Assignment blocks added to the branches of Selection block

On execution, the “Input” block would ask the user for an input and will proceed taking one of the branches of the “Selection” block depending on the condition whether “a <= 5” or not. In the example shown in Figure 17, the number 3 was read from the user, hence making the execution take the left “Yes” path from the Selection block.

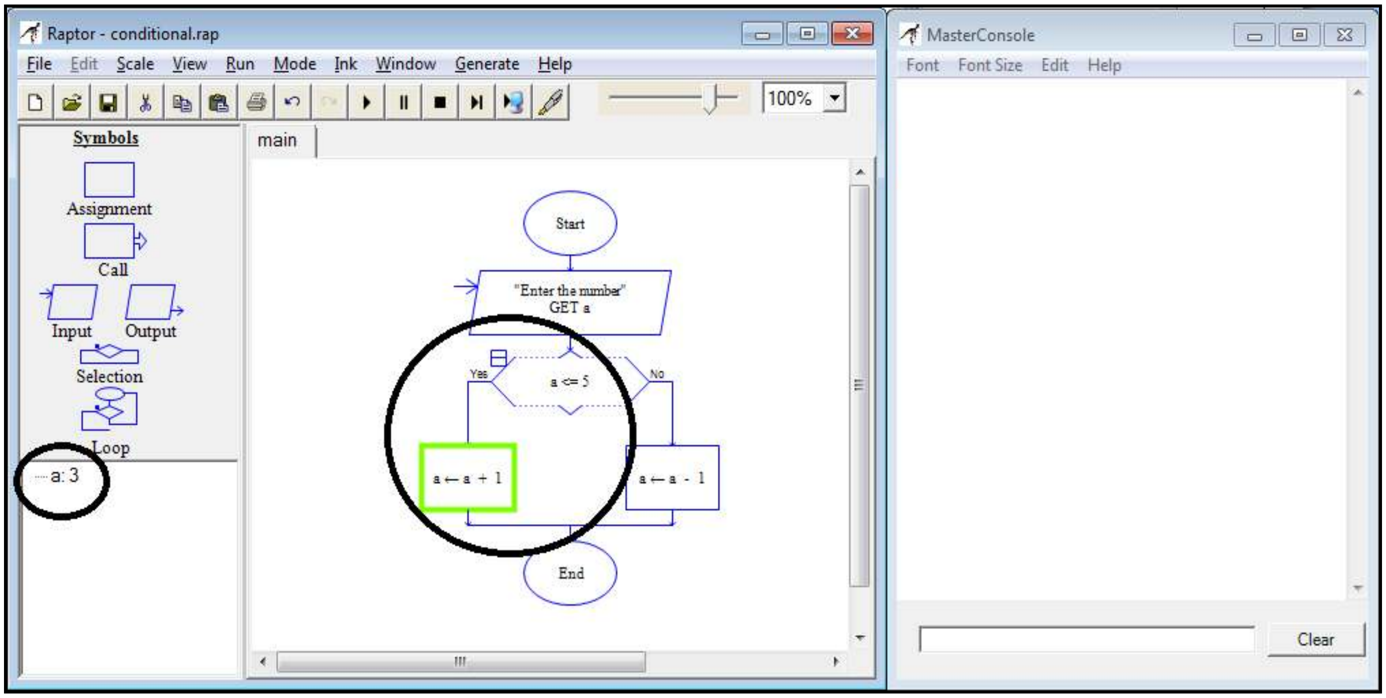
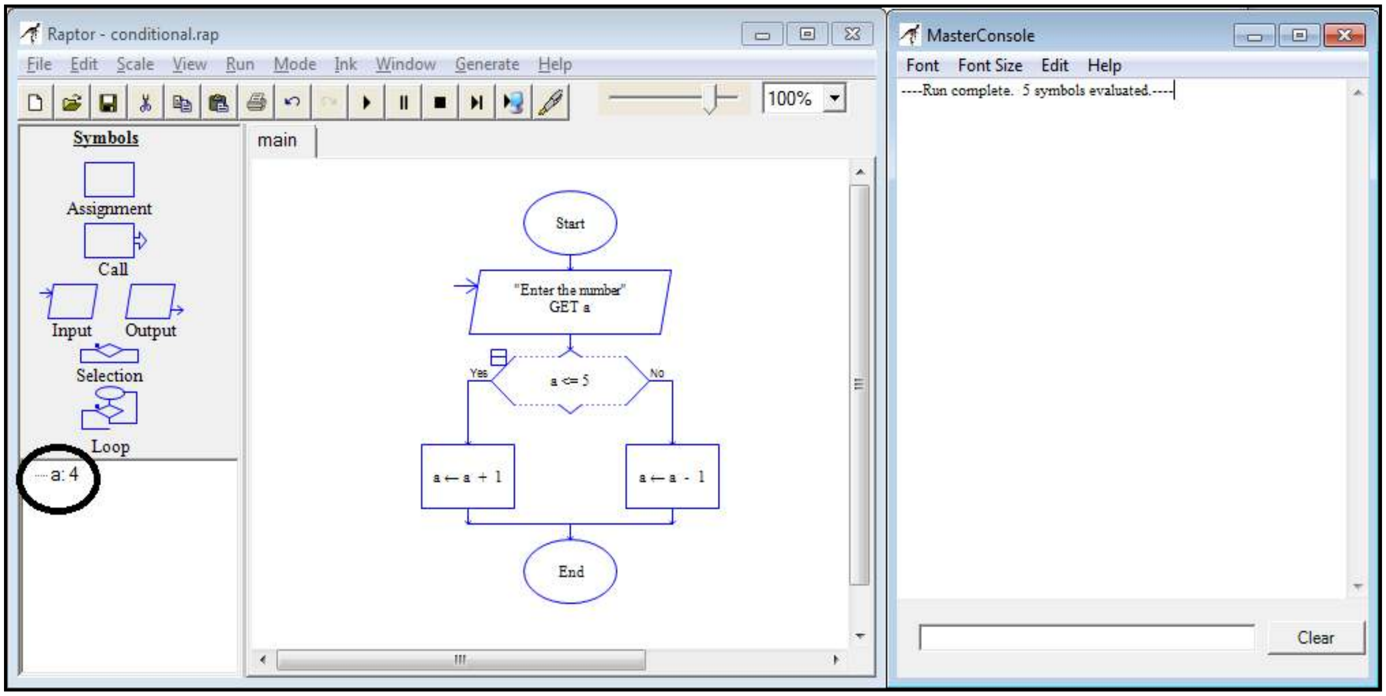


Figure 17: Selection block branching the execution according to the condition

After execution, the result is as shown in Figure 18; the “Watch Window” shows the variable “a”s value to be updated to a+1, i.e in this case, 4.



```
----Run complete.  5 symbols evaluated.----
```

Figure 18: Updated value after execution

1.3.6. Loops - Repeated execution of a block

Loops are used when we need a block of logic statements to run more than once. Loops logically require a termination condition which determines when the execution control moves out of the loop. In Raptor, loops are implemented using the "Loop" block in the "Symbols" section.

Let's take an example **(Example 3)** for illustrating the use of "Loop" block. We will try implementing a simple counting loop which will count from 1 to 10 and keep printing the number from inside the loop. A variable, say "a" which is initialized to "1" will be used for storing the values counted. The initialization steps until loop start is similar to the previous examples, so let's skip to the portion where the loop logic is to be added. As usual, click on the "Loop" block in "Symbols" and click on the part of the flowchart where we want to add it to. Once we do this, the window would look like Figure 19.

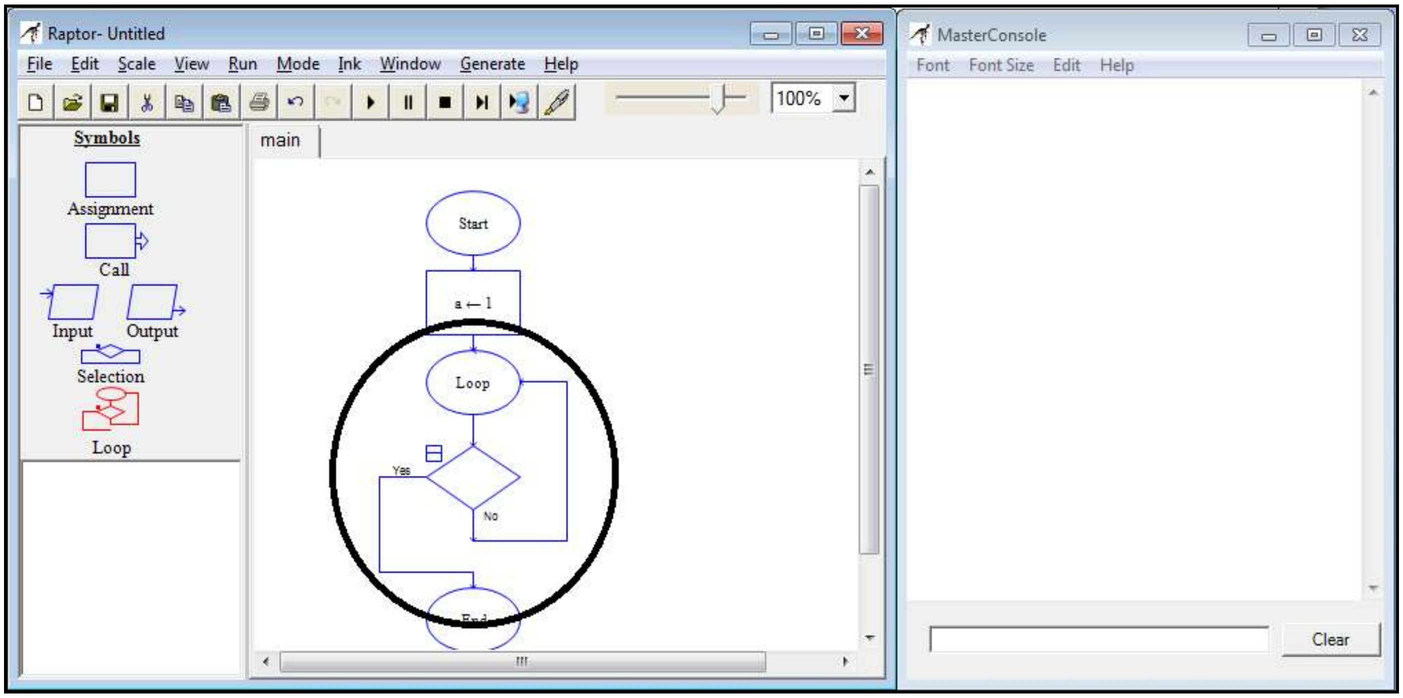


Figure 19: Adding a "Loop" block

As we can see from Figure 19, we have to configure the loop to check a condition on **NOT** satisfying which the loop proceeds to the next iteration. If the condition turns to be true, then the execution flow comes out of the loop.

For configuring the loop condition, we double click on the diamond box and a configuration window opens up as shown in Figure 20. We should enter the condition in the text box provided and press the "Done" button.

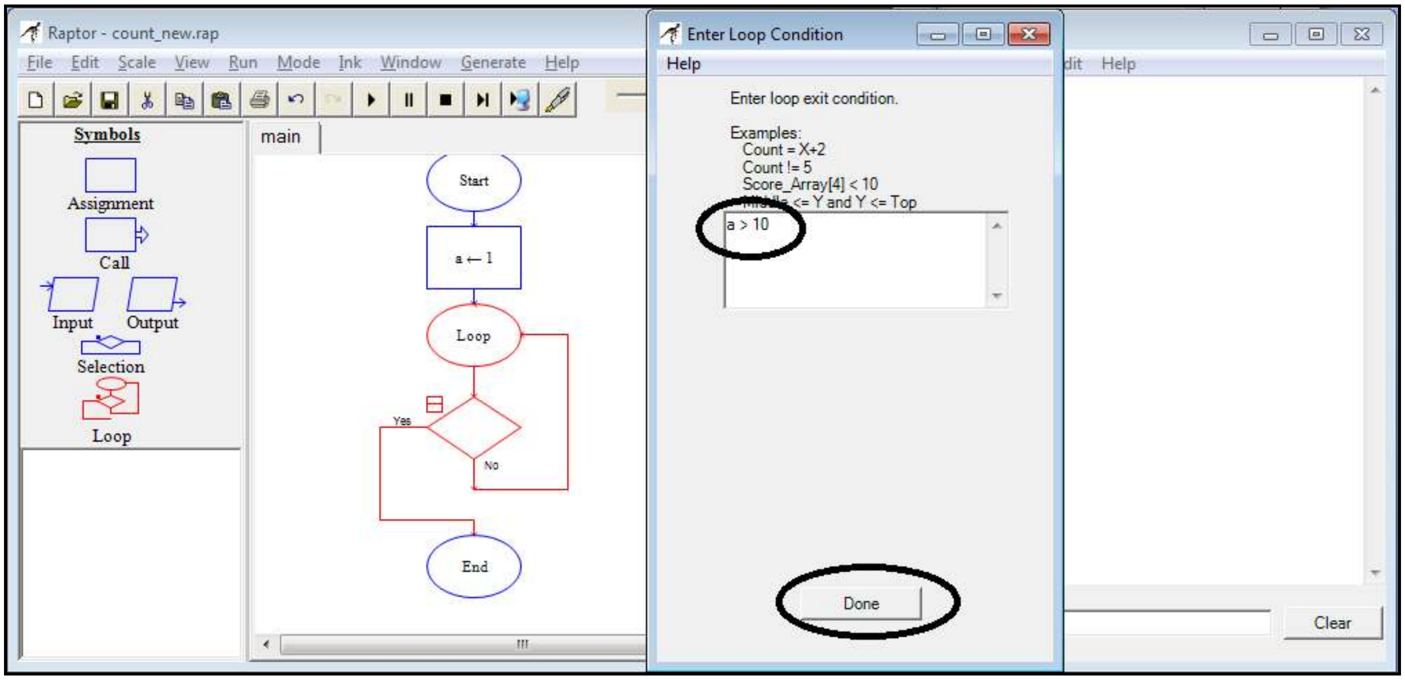


Figure 20: Configuring the “Loop” block

Now, we add the “Output” block to print the value of the variable “a” and the logic to increment the variable’s value inside the loop logic. This is shown in Figure 21.

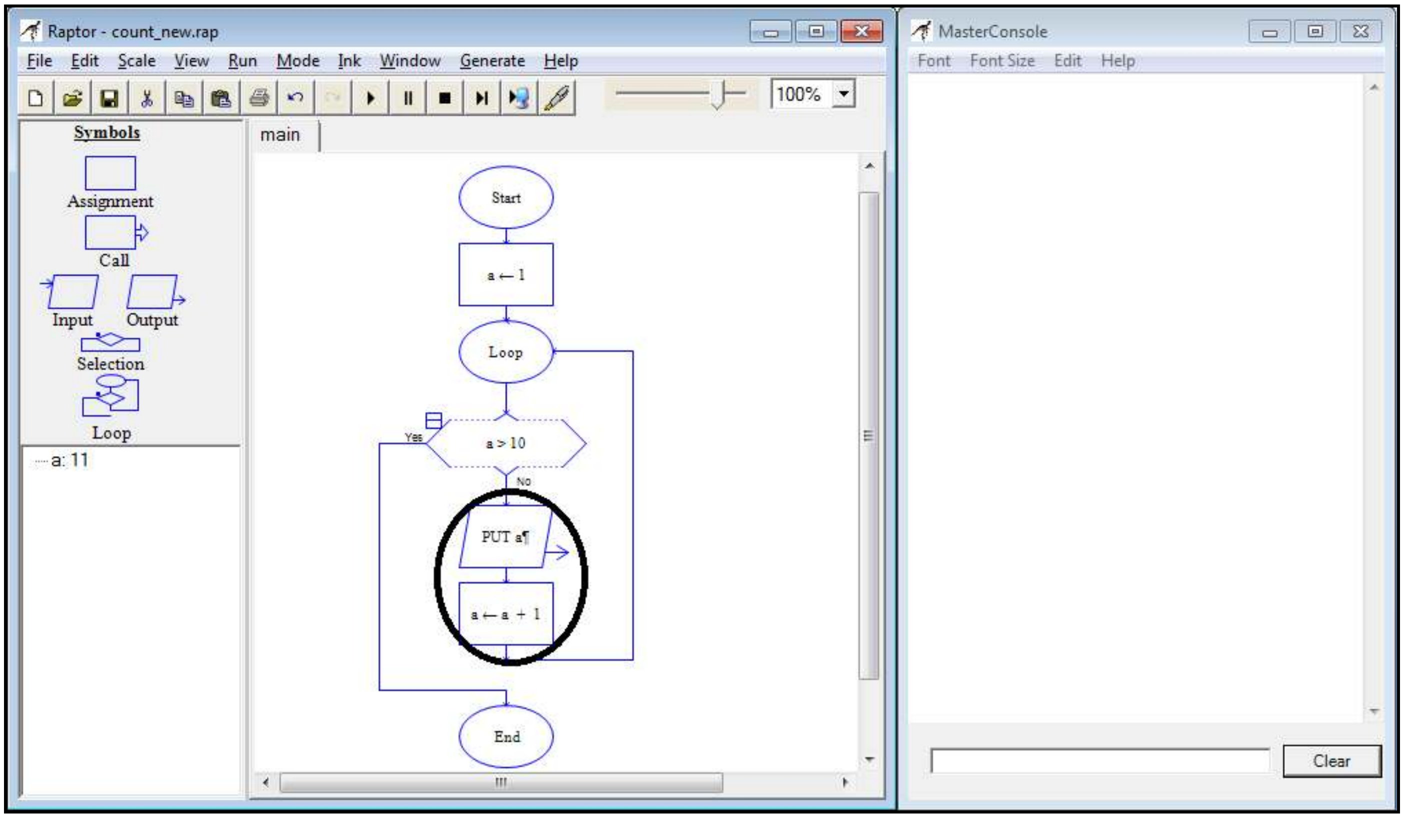
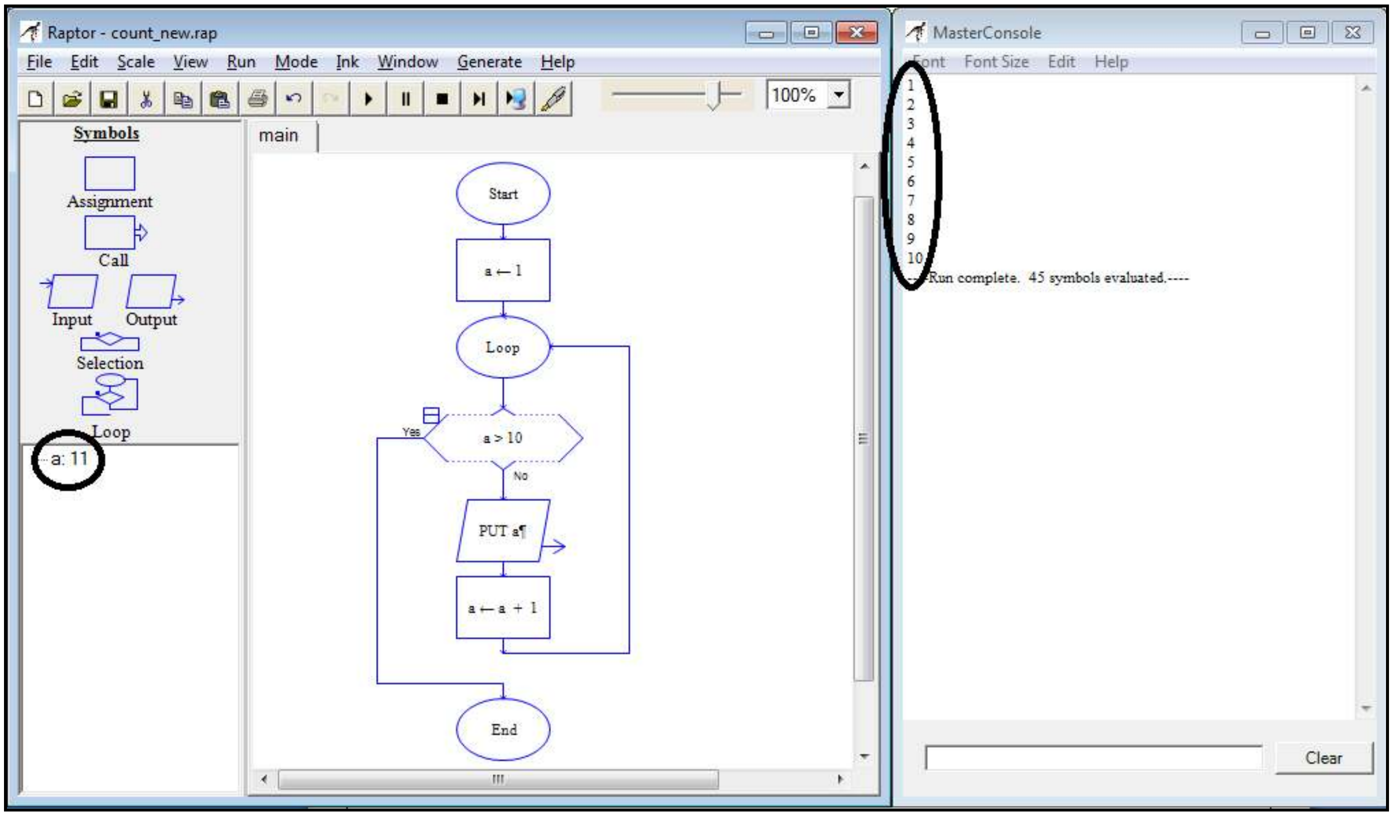


Figure 21: Adding the increment logic and “Output” block inside the loop construct

Once that is done, we can go ahead and execute the flowchart using the "Execute to Completion" button. We can see the results as shown in Figure 22.



```
a: 11
```

```
1
2
3
4
5
6
7
8
9
10
----Run complete.  45 symbols evaluated.----
```

Figure 22: The output of execution

Please note that in the "Watch Window" section, the value of the variable "a" has been updated to "11" which account to the last "increment" operation after which the control exited from the loop.

1.3.7. Functions

Prerequisite: Knowledge about modularity, abstraction and how functions work.

Functions are used when a set of logic blocks are designed for serving a specific well-defined task. The function can be "invoked" from the main flowchart along with passing some data to it, followed by the execution of the function and a "return" of a value from the function, but please note that both "data passing" and "returning" are not mandatory.

Let's take the **Example 3** explained above to illustrate how functions are implemented in Raptor. We will modify the same example to the "adding" part inside a function "add". Since the "input" parts are already explained in the Example 3 section, we will skip to the portion where "invoking" the function is done (generally called as a "function call"). Before that the flowchart will look as shown in Figure 23.

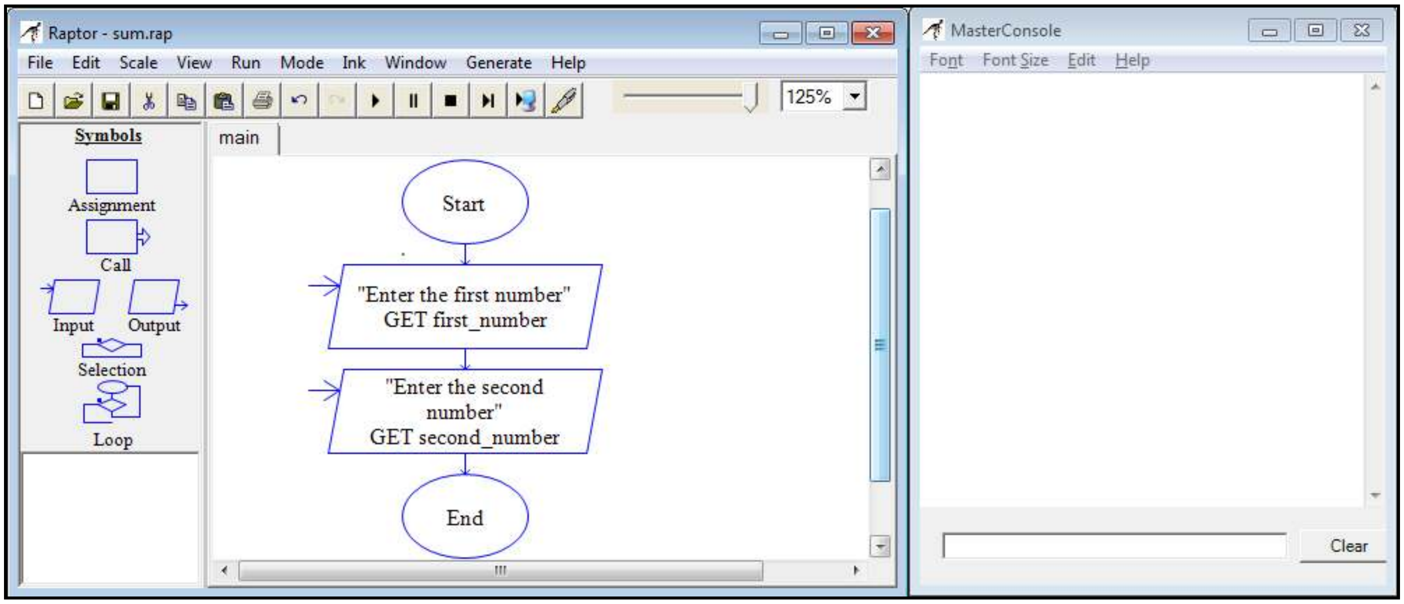


Figure 23: The flowchart before function call is added

Implementing the function call requires creating a new "Procedure" in the main flowchart. We do this by right clicking on the "main" tab and selecting "Add procedure", as shown in Figure 24.

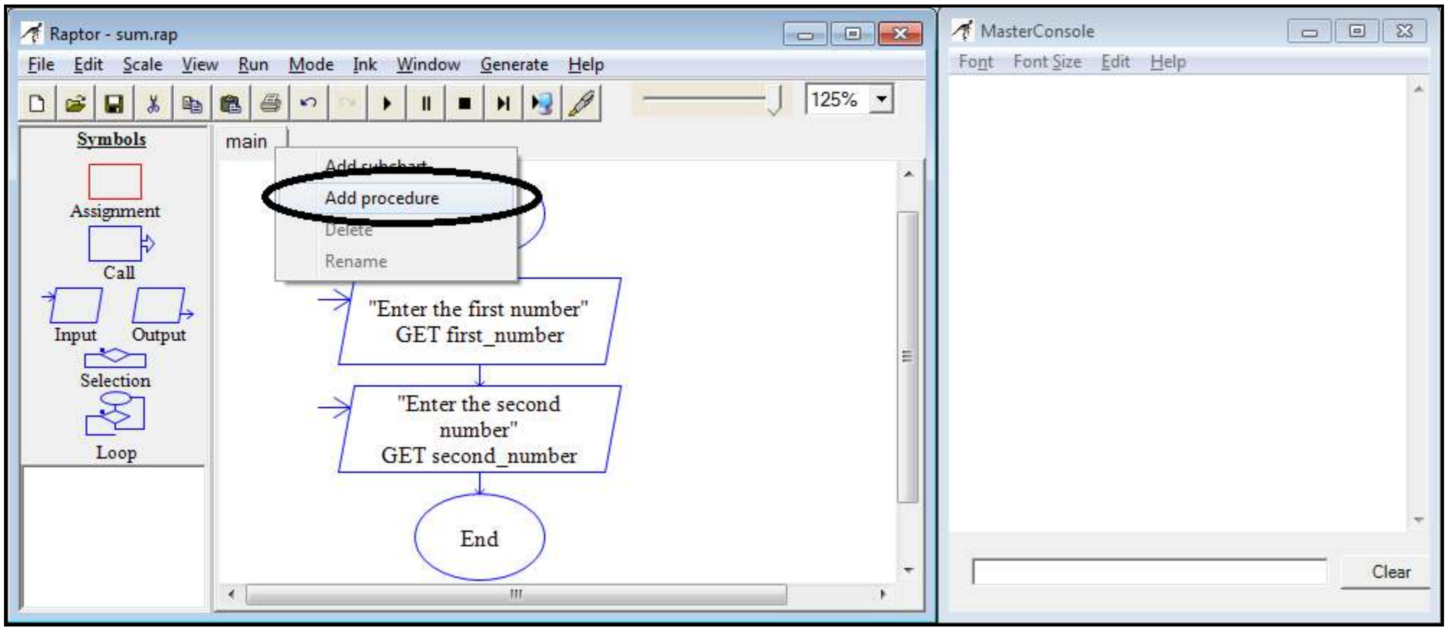


Figure 24: Adding a new function

Once we click on that option, a configuration window would open up where we enter the name of the function, the variables we intend to pass to it (Input checkbox should be checked) and the variable that the function is supposed to return (Output checkbox should be checked), as shown in Figure 25. Once this is done, click on the "OK" button.

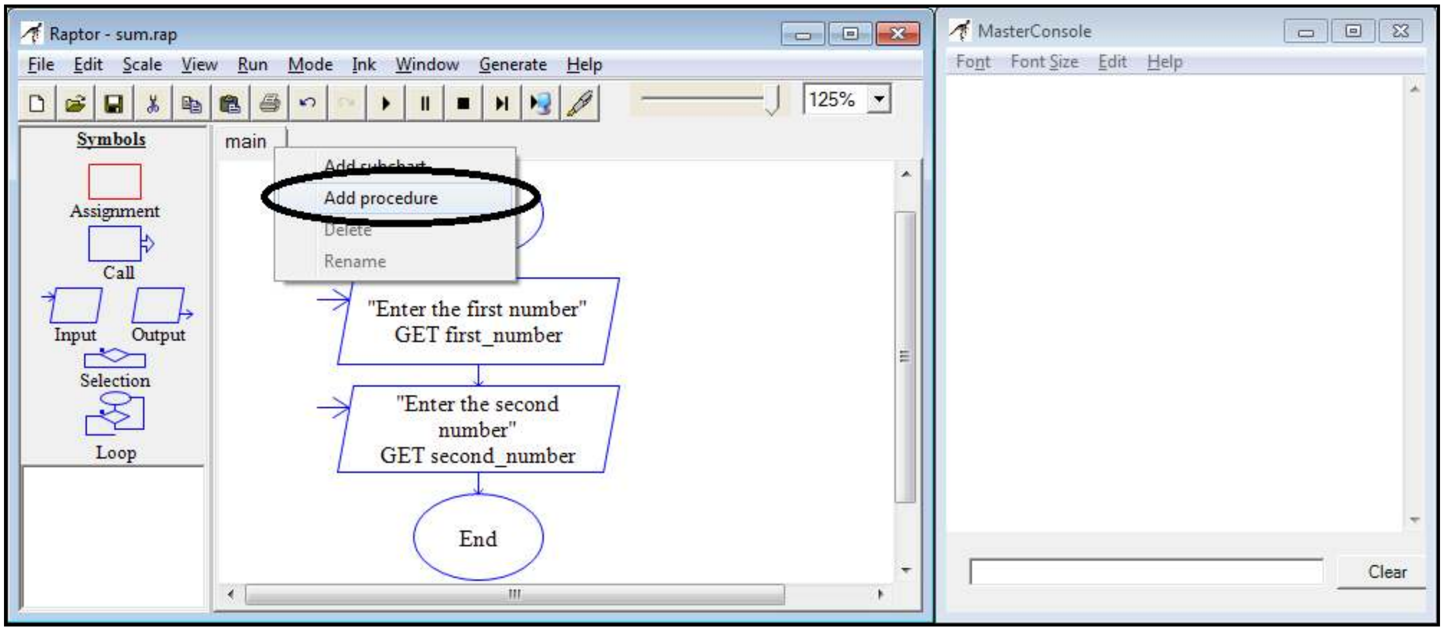


Figure 25: Configuration the function

A new tab having the function name we gave ("add") would be displayed with parameters mentioned in the "Start" block. Please see Figure 26.

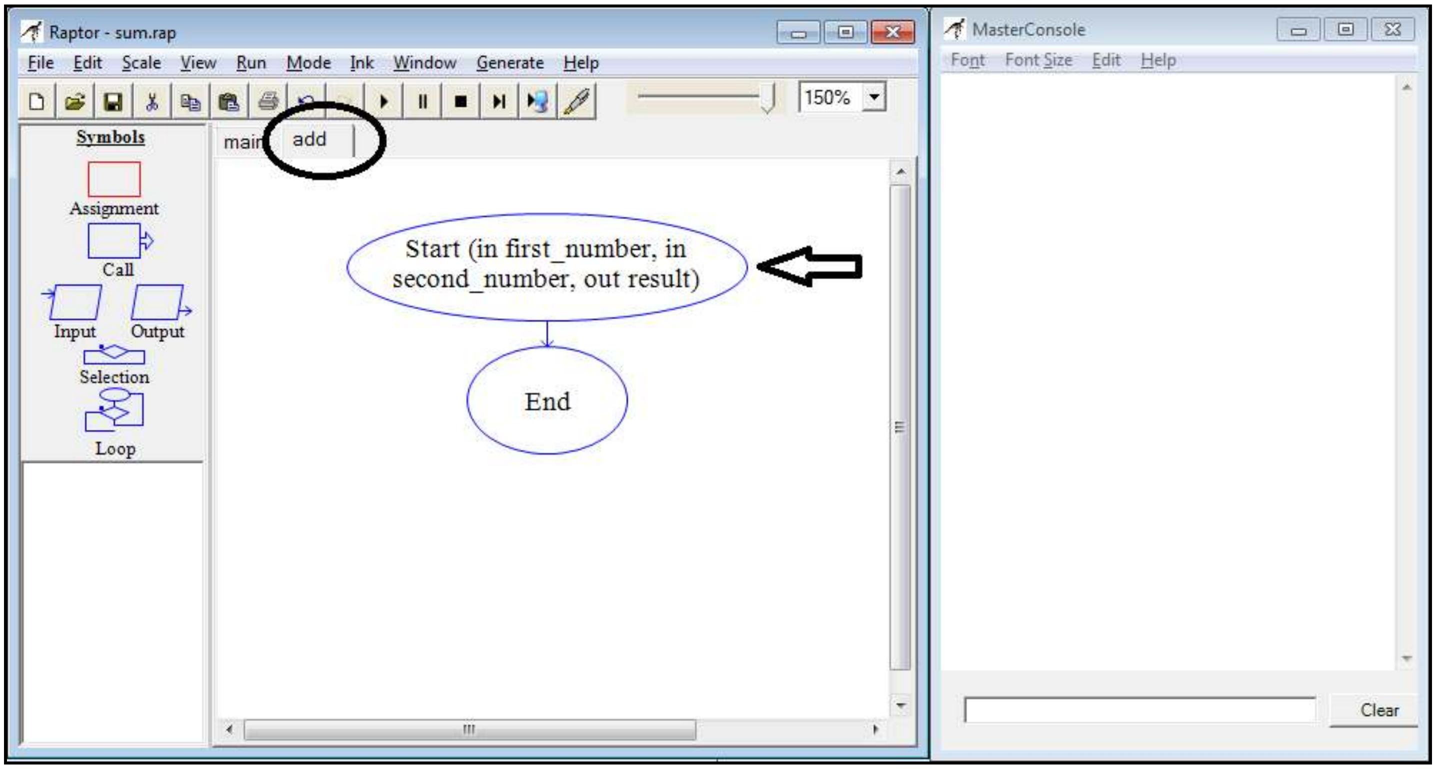
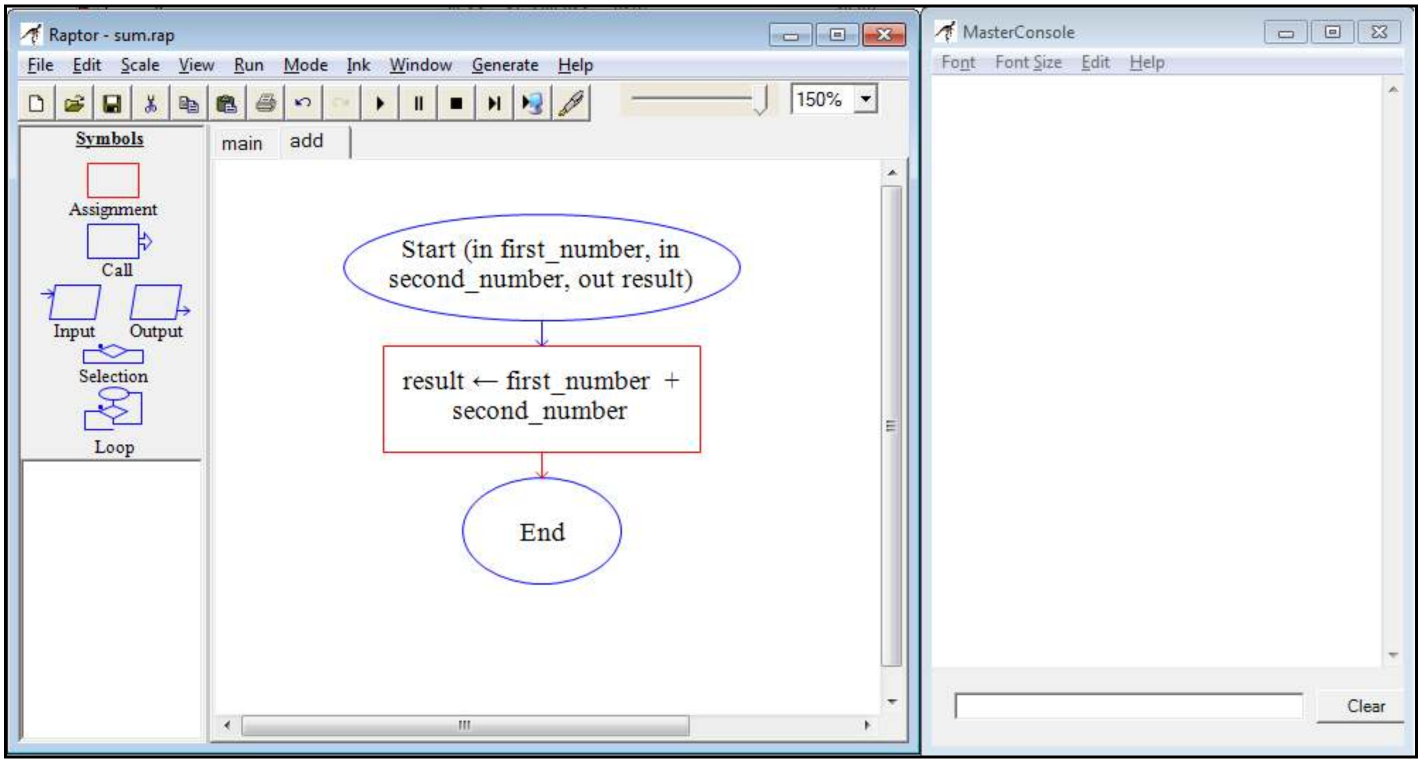


Figure 26: Function open in separate tab

Now, we add the logic to add the two numbers and to assign the sum into the variable "result". This is to be done with the help of "Assignment" block in the "Symbols" section. Once this is done, the flowchart would look like Figure 27.



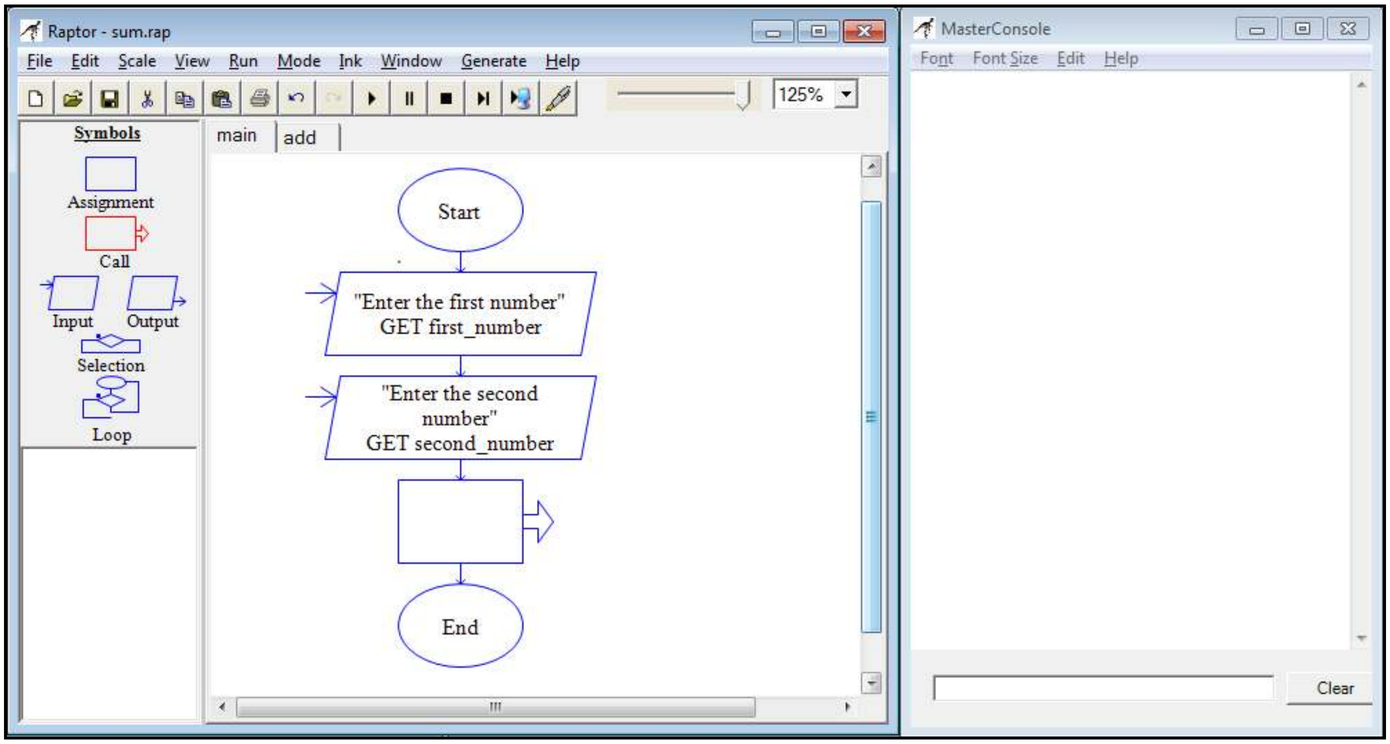
Start (in first_number, in second_number, out result)

result ← first_number + second_number

End

Figure 27: Adding logic inside function

Now, we should go back to the "main" tab and add a call to invoke the function "add". This is done using the "Call" block from the "Symbols" section which we add to the flowchart as shown in Figure 28.



Start

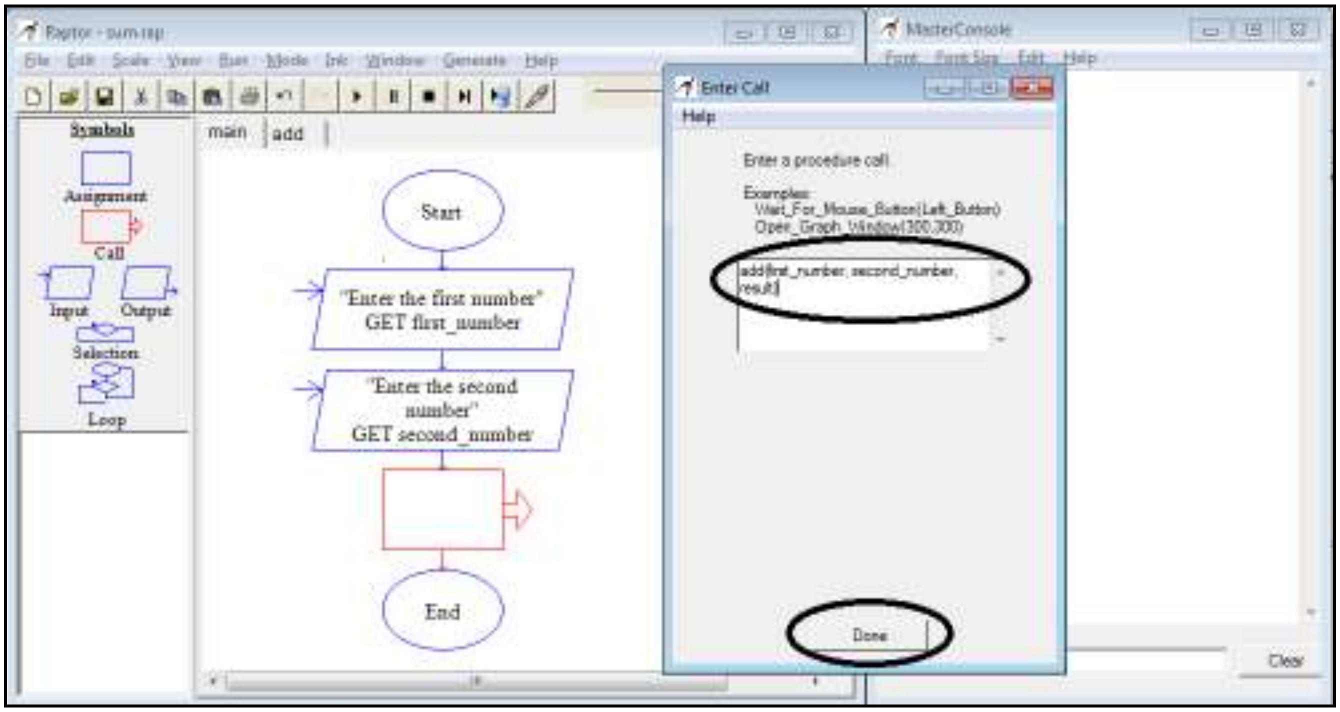
"Enter the first number"
GET first_number

"Enter the second number"
GET second_number

End

Figure 28: Adding the "Call" block to invoke function

Now, we configure the “Call” block to call the function “add” along with the parameters to be passed to it which in our example is “first_number” and “second_number”. We do this by double clicking the “Call” block and filling these details in the text boxes available and clicking the “Done” button, as shown in Figure 29.


Raptor - sum.rap
File Edit Scale View Run Mode Ink Window Generate Help
Symbols
Assignment
Call
Input
Output
Selection
Loop
main
add
Start
"Enter the first number" GET first_number
"Enter the second number" GET second_number
End
Enter Call
Help
Done
MasterConsole
Font Font Size Edit Help
Clear

Enter a procedure call

Examples:
Wait_For_Mouse_Button(Left_Button)
Open_Graph_Window(300,300)

add(first_number, second_number, result)

Figure 29: Configuring the “Call” block to invoke function “add” with parameters

After the configuration is done, the flowchart would look as shown in Figure 30.

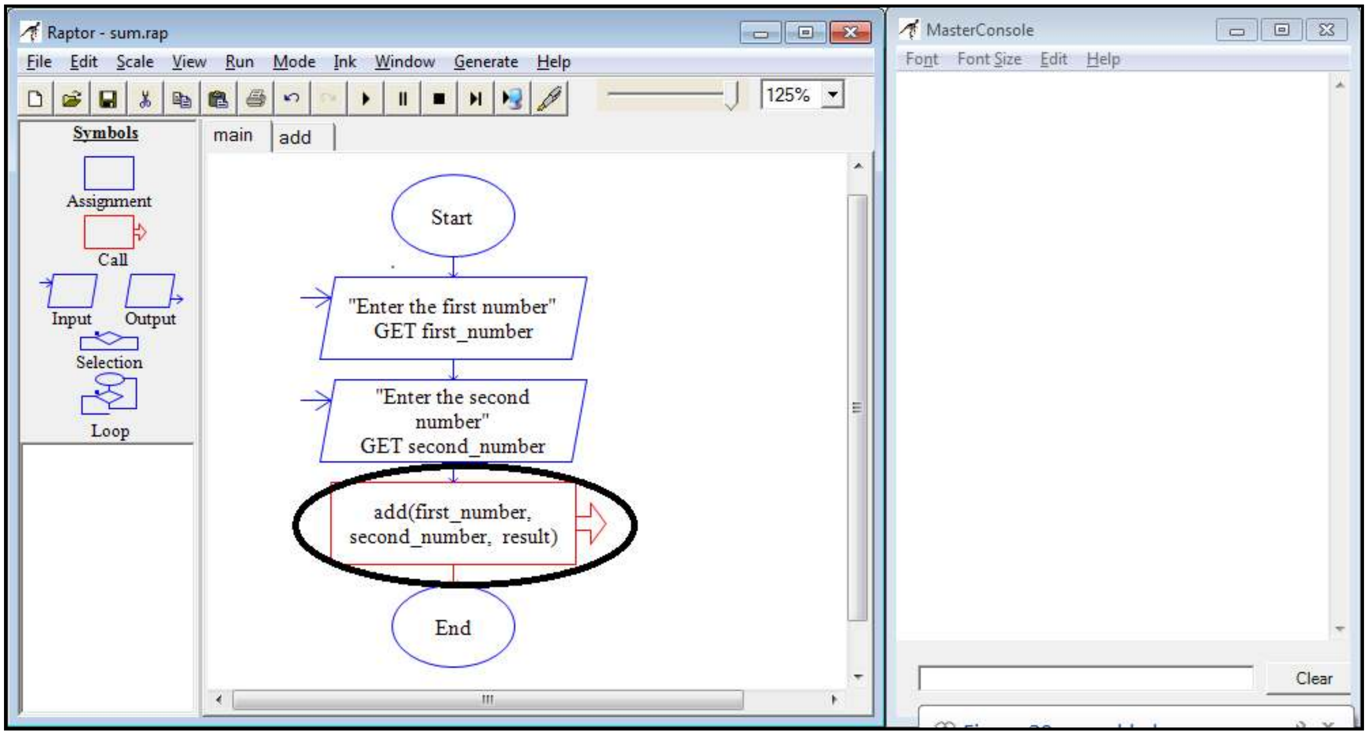

Raptor - sum.rap
File Edit Scale View Run Mode Ink Window Generate Help
125%
Symbols
Assignment
Call
Input
Output
Selection
Loop
main
add
Start
"Enter the first number" GET first_number
"Enter the second number" GET second_number
add(first_number, second_number, result)
End
MasterConsole
Font Font Size Edit Help
Clear

Figure 30: Configuring the “Call” block to invoke function “add” with parameters

Now, we add an "Output" block to print the output returned by the function. Please refer Figure.31.

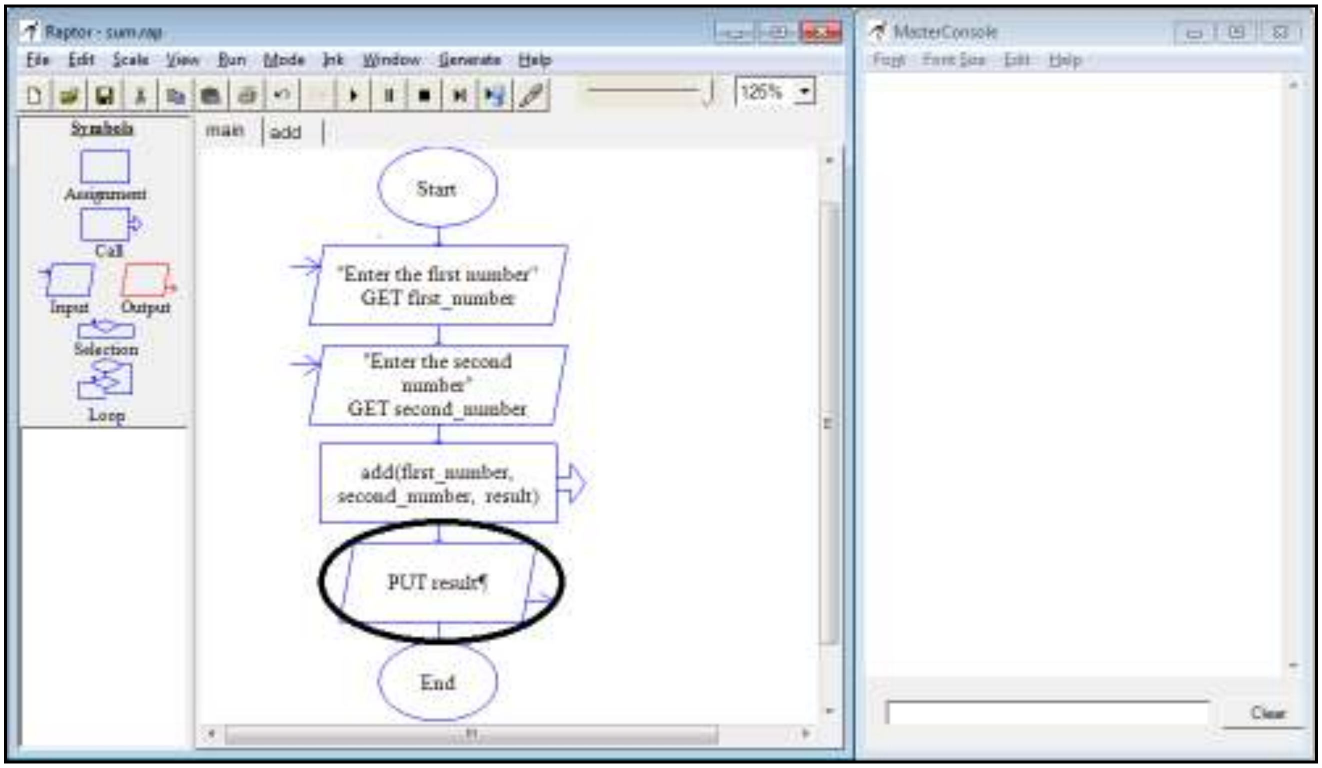
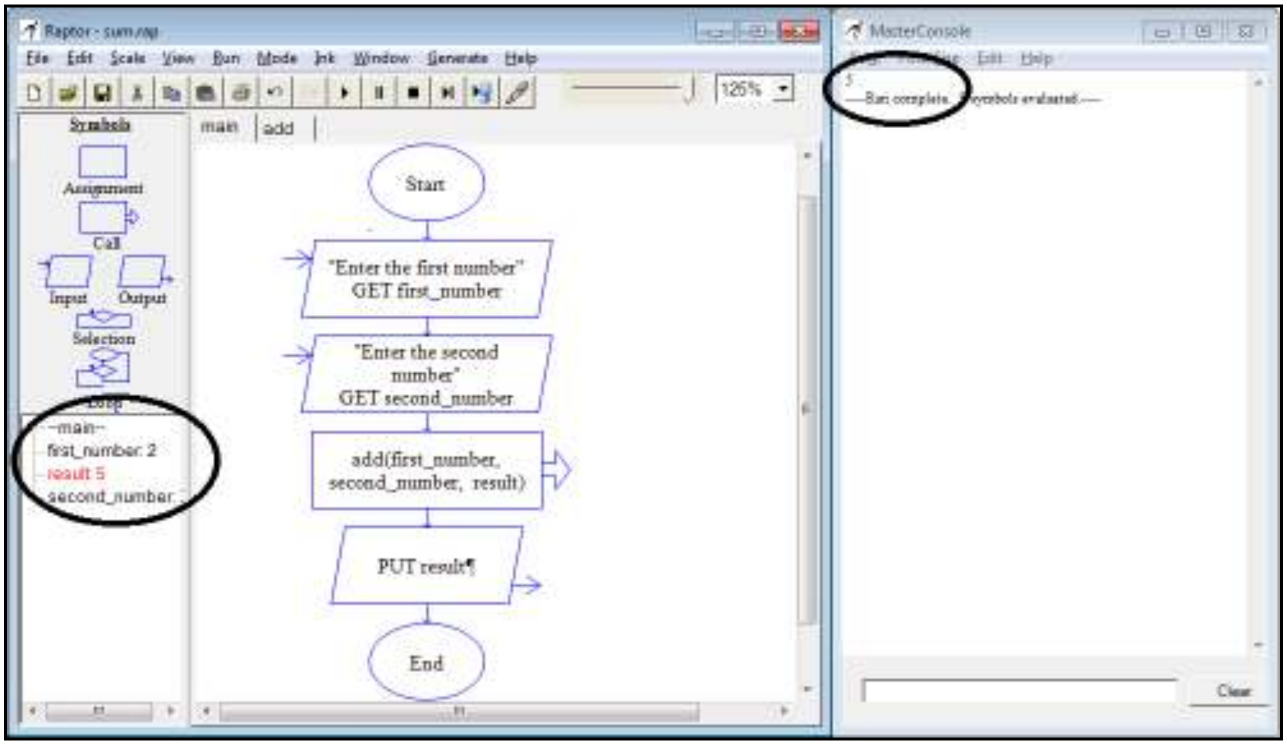


Figure 31: "Output" block added to the flowchart

Now, we can go ahead and run the program. The control flow would pass to the "add" tab and return to the "main" tab during execution. The results would be as shown in Figure 32.



```
--main--
first_number: 2
result: 5
second_number: 
```

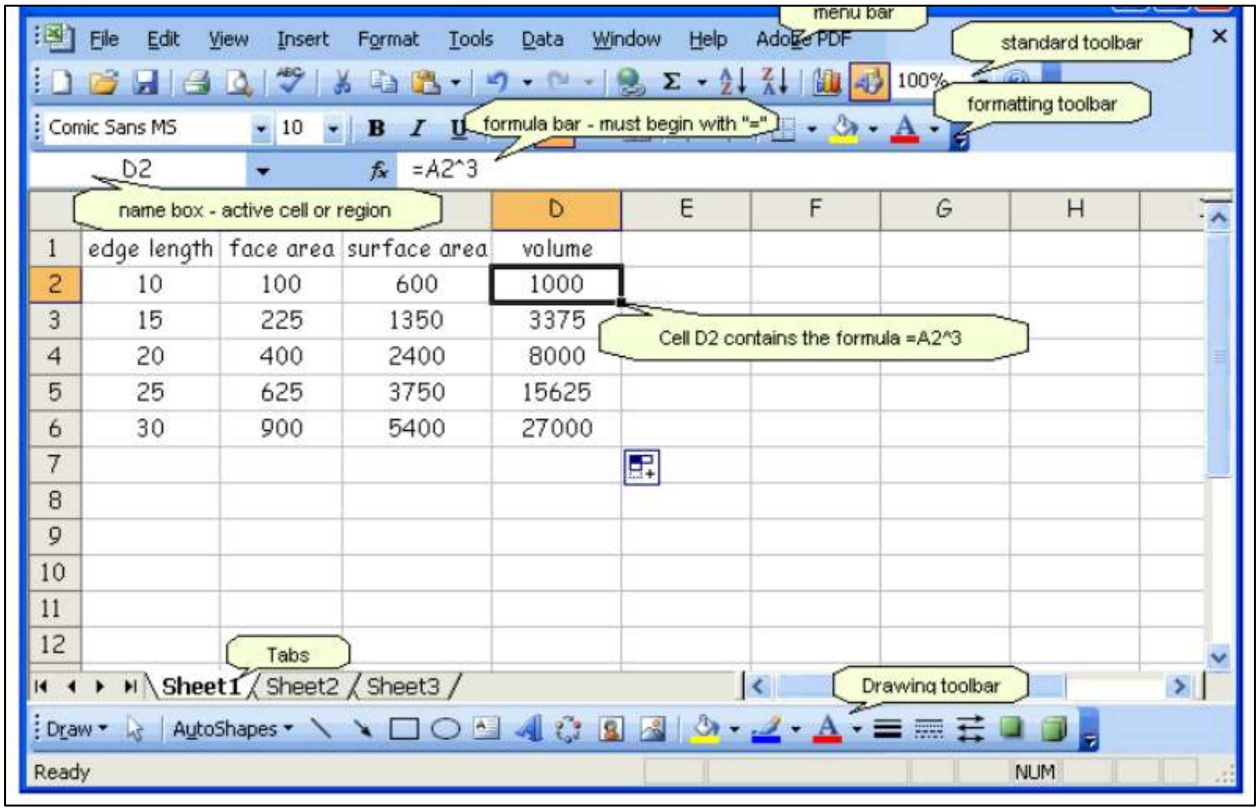
```
5
----Run complete. symbols evaluated.----
```

Figure 32: The output of execution

2. INFORMATION AND DATA

2.1. Data Handling in Spread Sheets

Start Excel and a screen similar to the one below will appear. The main body of the screen will contain columns labeled with letters (A, B, C...) and rows labeled with numbers (1, 2, 3...). A cell is a space where numbers, formulae or text can be entered. Each cell has a designation that gives its location such as F22 (column F, row 22) or CC115 (column CC, row 115).


File Edit View Insert Format Tools Data Window Help Adobe PDF
menu bar
standard toolbar
formatting toolbar
Comic Sans MS 10
formula bar - must begin with "="
name box - active cell or region
Cell D2 contains the formula =A2^3
Tabs
Sheet1 Sheet2 Sheet3
Drawing toolbar
Draw AutoShapes
Ready
NUM

D2 =A2^3

	A	B	C	D	E	F	G	H
1	edge length	face area	surface area	volume				
2	10	100	600	1000				
3	15	225	1350	3375				
4	20	400	2400	8000				
5	25	625	3750	15625				
6	30	900	5400	27000				
7								
8								
9								
10								
11								
12								

Above the main body are two white regions. The left region, the Name box, indicated which cell or region (such as a chart) is active (selected). The right region is a Formula bar. Cell contents are displayed here. If you activate a cell that has a number generated by a formula, the value will appear in the cell but the formula will appear in the formula bar.

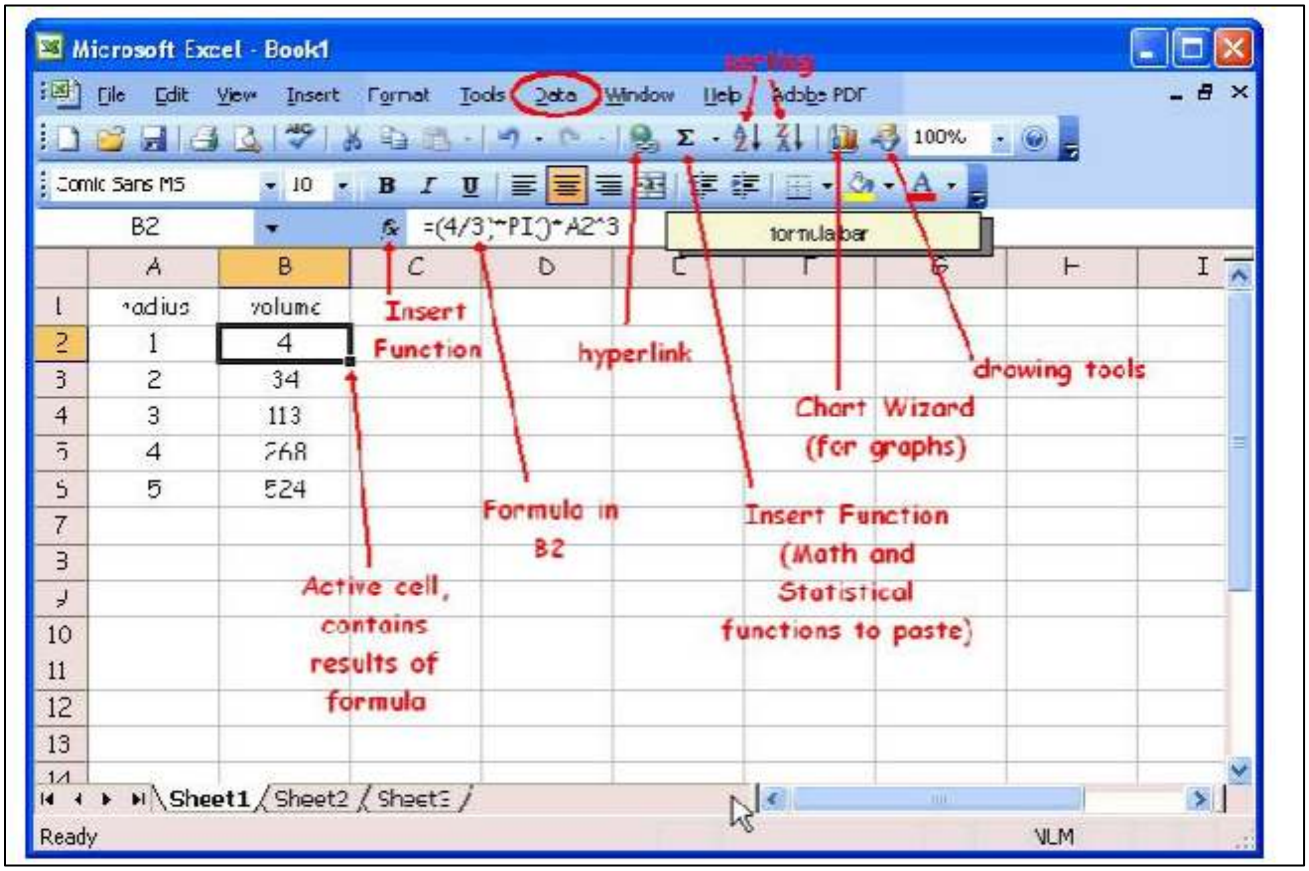
You enter data into a cell by placing the cursor in the cell, clicking, and typing. You will notice that what you type appears in the cell and in the formula bar. Data is registered by hitting Enter or using any cursor key to change cell location. If you need to edit the contents of a cell, click on the cell and then on the formula bar; you can then change the entry in the cell or in the formula bar. Register as before.

Above the Name box and Formula bar are two rows of icons called the Formatting and Standard Toolbars. Some of the icons will resemble ones you have seen in word processors and others are unique to spreadsheets. If you place the cursor on an icon and wait a few seconds, the name or function of the icon will be displayed. This tool bar can be customized with additional features you frequently use by going to View/Toolbars/Customize. Above the tool bar is the Menu bar. Each item here has a pull-down menu of operations. Note that the menu item that reads Data on the menu bar (on the graphic above) changes to Chart when you have a chart active.

If you wish to format a whole column or row of cells, click on the column letter or row number and the entire column or row will be highlighted. Any change you make, such as defining the numerical display (under Format) or deleting, will apply to all of the highlighted cells. If you right-click the letter or number (or any cell) you will get an abbreviated menu of common operations.

Look at the sample sheet. You will notice that cell D2 contains the formula =A2^3. If you copy this formula and paste it into D3, Excel will use the value in cell A3 rather than A2 for the calculation. If you paste into cell D4 it will use A4. This is great if you want to extend a calculation for a whole series of data.

Bring up Excel on the computer and a screen like the one below will appear. This is called a "worksheet" in Excel and since it is active, the word "Data" appears as a pull-down menu. Many of the toolbar icons are similar to ones used in word processing programs. Some of the ones more useful in science and math are labeled below.

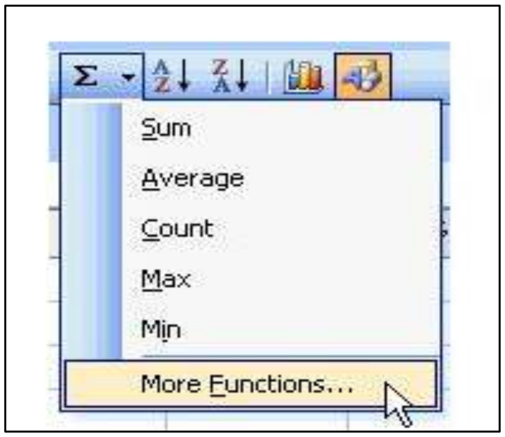


B2 =(4/3)*PI()*A2^3

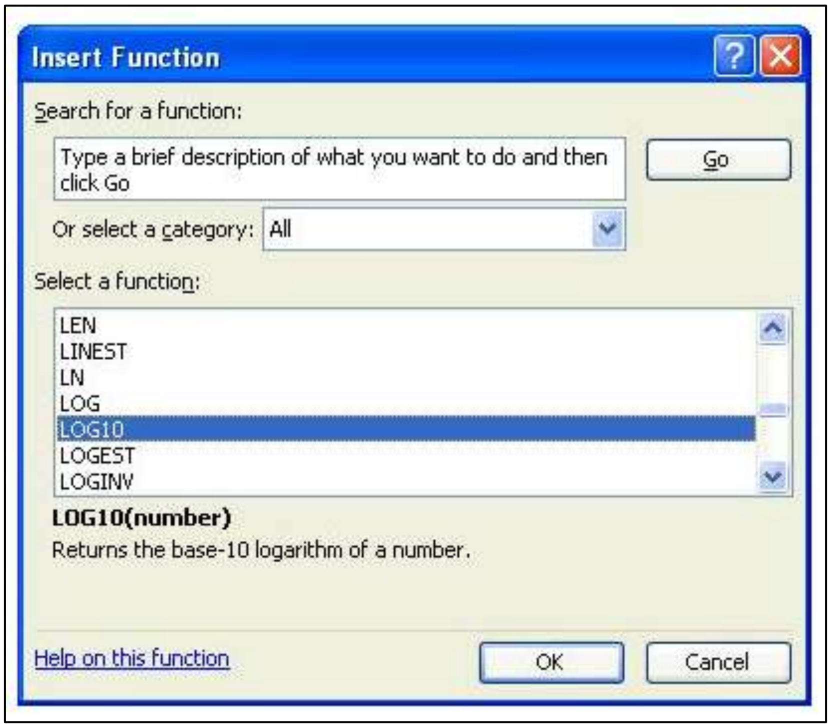
	A	B
1	radius	volume
2	1	4
3	2	34
4	3	113
5	4	268
6	5	524

The first task is to enter your data into the columns. For ease of graphing, try to place your independent variable in the first column. You may want to label the first cell in each column (A1, B1, C1, etc.) with the variable names.

Insert Functions: Using the Insert Function (fx icon) or from the toolbar, which lists some common functions as shown below, a vast number of mathematical and statistical functions can be placed into a formula.



When you select a function, a brief explanation with syntax is given in the lower part of this screen as seen below on the Insert Function pop-up screen below.



LOG10(number)
Returns the base-10 logarithm of a number.

This screen shows the common logarithm function under Math.

Further explanation of the function is available by clicking on the “Help on this function.”

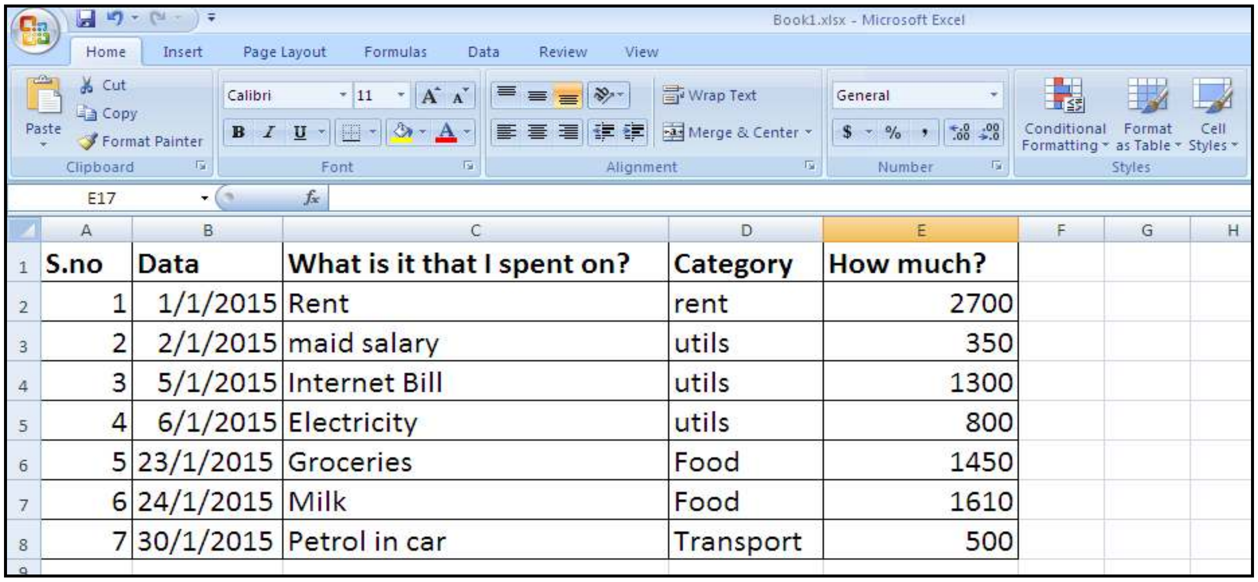
Select the function and click OK. This will bring up another pop-up window with instructions to follow. Or you can type the function name in the formula bar directly.

Remember to follow the syntax. Square root, raise to a power, average, and standard deviation are a few other examples of available operations.

2.2. Data handling – Using Formula Bar

Enter the following data in the Excel sheet maintained for the daily expenses in a family for a Month. Use the same data for all the formulae.

Sample spread sheet



S.no	Data	What is it that I spent on?	Category	How much?
1	1/1/2015	Rent	rent	2700
2	2/1/2015	maid salary	utils	350
3	5/1/2015	Internet Bill	utils	1300
4	6/1/2015	Electricity	utils	800
5	23/1/2015	Groceries	Food	1450
6	24/1/2015	Milk	Food	1610
7	30/1/2015	Petrol in car	Transport	500

2.2.1. Data handling - Sum function

Explanation

To add a set of given values in the spread sheet

Formula in the spread sheet:

SUM(number 1,[number 2], …[number n]) or

SUM(cell location of number 1 : cell location of number n)

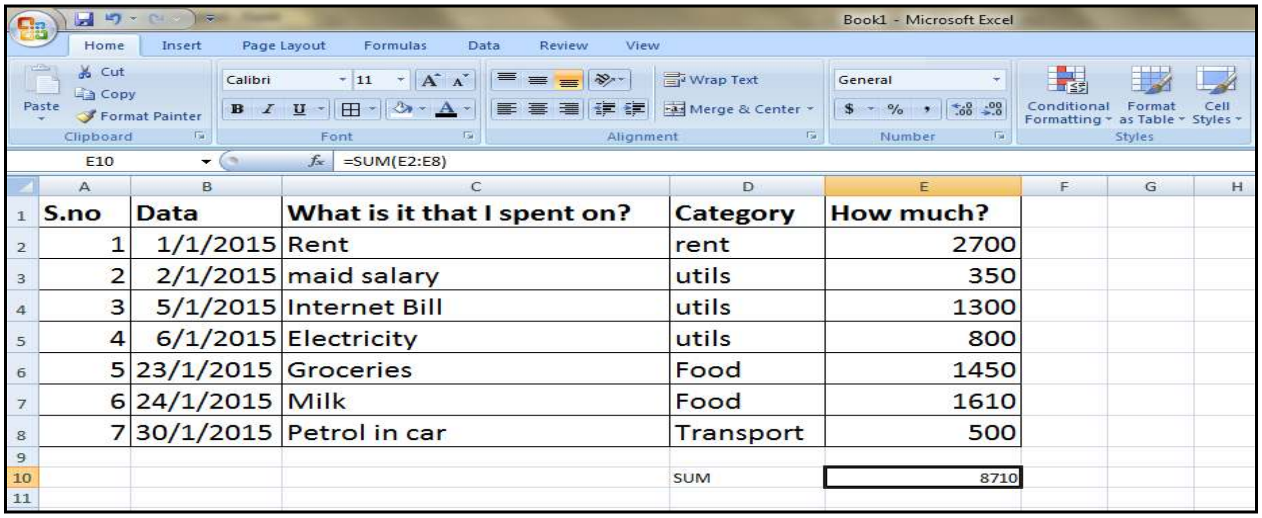
Where number1, number2, number n are the values to be summed

Example: To add the set of values given in the sample sheet (How much spent (column E))

Steps:

1. Enter the data as shown in the above sample spread sheet.
2. In D10 location types the text SUM.
3. In E10 type =SUM(E2:E8) and press enter key.
4. The result displayed in E10 is 8710

Screen shot:



E10 =SUM(E2:E8)

	A	B	C	D	E
1	S.no	Data	What is it that I spent on?	Category	How much?
2	1	1/1/2015	Rent	rent	2700
3	2	2/1/2015	maid salary	utils	350
4	3	5/1/2015	Internet Bill	utils	1300
5	4	6/1/2015	Electricity	utils	800
6	5	23/1/2015	Groceries	Food	1450
7	6	24/1/2015	Milk	Food	1610
8	7	30/1/2015	Petrol in car	Transport	500
9					
10				SUM	8710
11					

2.2.2. Data handling – Average function

Explanation

To calculate the average of given values in the spread sheet

Formula in the spread sheet:

AVERAGE(number 1,[number 2], …[number n]) or

AVERAGE(cell location of number 1 : cell location of number n)

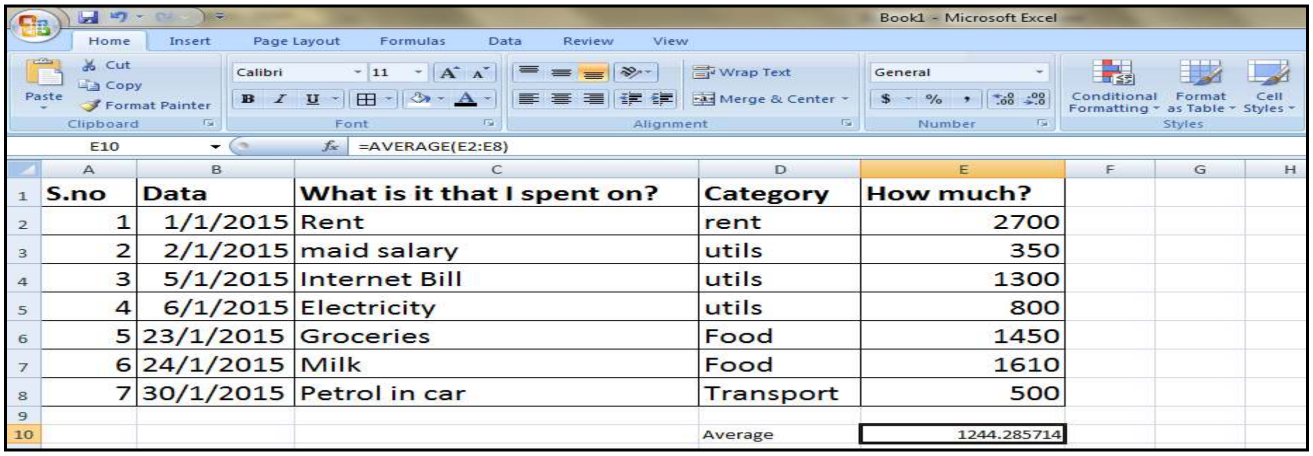
Where number1, number2, number n are the values used to find the average

Example: To average the set of values given in the sample sheet (How much spent (column E))

Steps:

1. Enter the data as shown in the above sample spread sheet.
2. In D10 location types the text Average.
3. In E10 type =AVERAGE(E2:E8) and press enter key.
4. The result displayed in E10 is 1244.285714

Screen shot:



E10 =AVERAGE(E2:E8)

	A	B	C	D	E
1	S.no	Data	What is it that I spent on?	Category	How much?
2	1	1/1/2015	Rent	rent	2700
3	2	2/1/2015	maid salary	utils	350
4	3	5/1/2015	Internet Bill	utils	1300
5	4	6/1/2015	Electricity	utils	800
6	5	23/1/2015	Groceries	Food	1450
7	6	24/1/2015	Milk	Food	1610
8	7	30/1/2015	Petrol in car	Transport	500
9					
10				Average	1244.285714

2.2.3. Data handling – Count Function

Explanation

To count a set of given values in the spread sheet

Formula in the spread sheet:

COUNT(number 1,[number 2], …[number n]) or

COUNT(cell location of number 1 : cell location of number n)

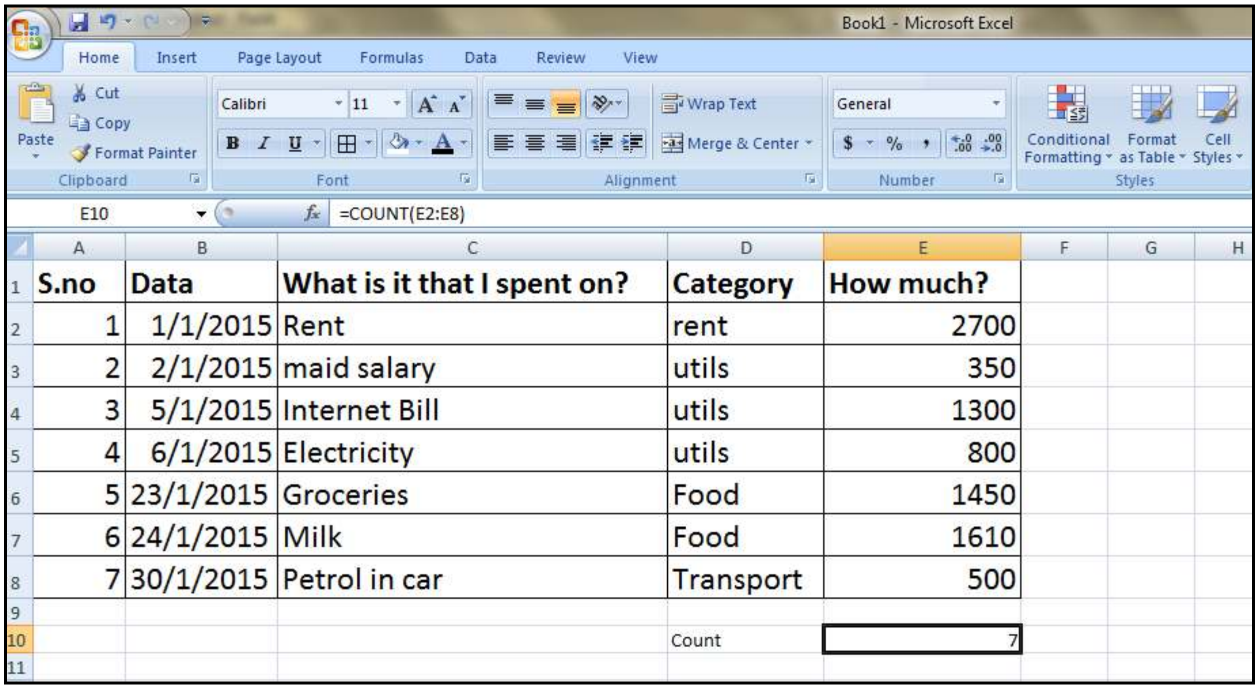
Where number1, number2, number n are the values to be counted

Example: To count the set of values given in the sample sheet (How much spent (column E))

Steps

1. Enter the data as shown in the above sample spread sheet.
2. In D10 location types the text Count.
3. In E10 type =COUNT(E2:E8) and press enter key.
4. The result displayed in E10 is 7

Screen shot:



=COUNT(E2:E8)

	A	B	C	D	E
1	S.no	Data	What is it that I spent on?	Category	How much?
2	1	1/1/2015	Rent	rent	2700
3	2	2/1/2015	maid salary	utils	350
4	3	5/1/2015	Internet Bill	utils	1300
5	4	6/1/2015	Electricity	utils	800
6	5	23/1/2015	Groceries	Food	1450
7	6	24/1/2015	Milk	Food	1610
8	7	30/1/2015	Petrol in car	Transport	500
9					
10				Count	7
11					

2.2.4. Data handling – Maximum function

Explanation

To find the maximum of a set of given values in the spread sheet

Formula in the spread sheet:

MAX(number 1,[number 2], …[number n]) or

MAX(cell location of number 1 : cell location of number n)

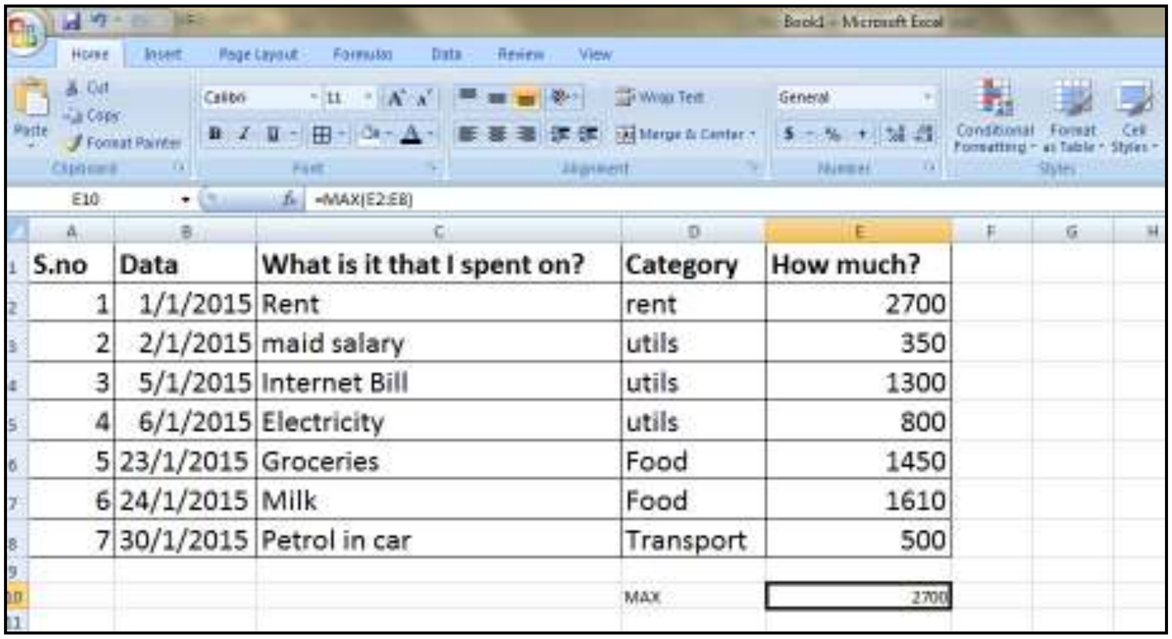
Where number 1, number 2, number n are the values used to find the maximum

Example: To find the maximum of the given values in the sample sheet (How much spent (column E))

Steps

1. Enter the data as shown in the above sample spread sheet.
2. In D10 location types the text MAX.
3. In E10 type =MAX(E2:E8) and press enter key.
4. The result displayed in E10 is 2700

Screen shot:



E10 =MAX(E2:E8)

	A	B	C	D	E
1	S.no	Data	What is it that I spent on?	Category	How much?
2	1	1/1/2015	Rent	rent	2700
3	2	2/1/2015	maid salary	utils	350
4	3	5/1/2015	Internet Bill	utils	1300
5	4	6/1/2015	Electricity	utils	800
6	5	23/1/2015	Groceries	Food	1450
7	6	24/1/2015	Milk	Food	1610
8	7	30/1/2015	Petrol in car	Transport	500
9					
10				MAX	2700

2.2.5. Data handling – Minimum function

Explanation

To find the minimum of a set of given values in the spread sheet

Formula in the spread sheet:

MIN(number 1,[number 2], …[number n]) or

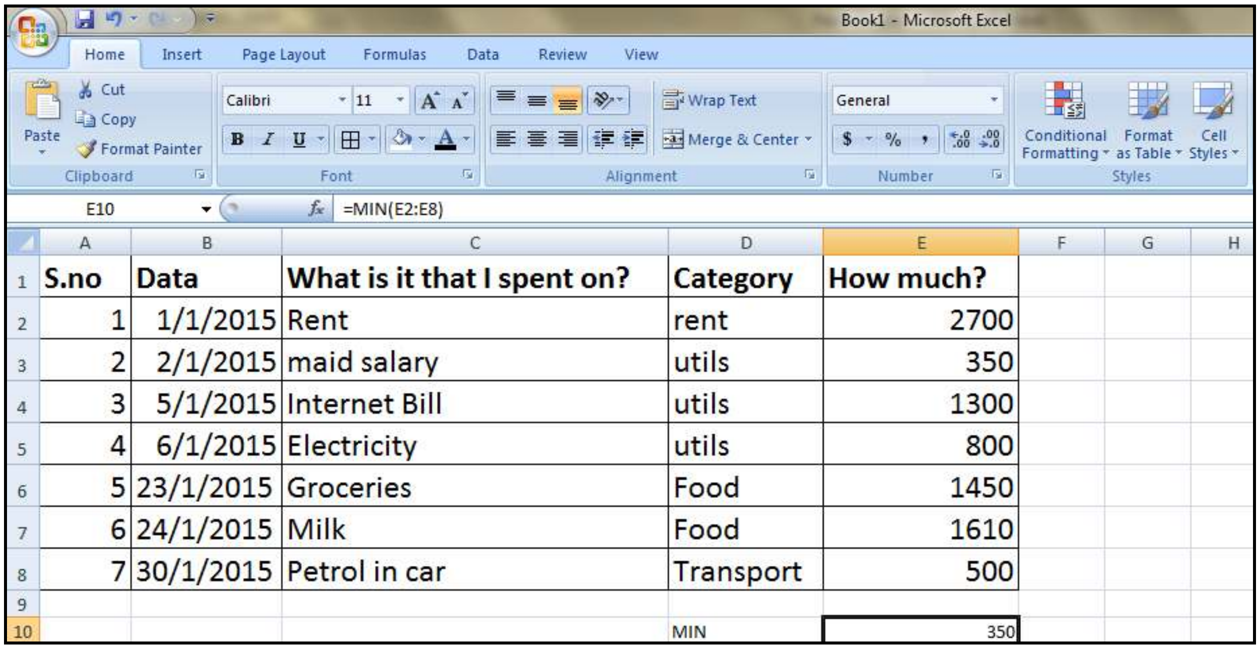
MIN(cell location of number 1 : cell location of number n)

Where number 1, number 2, number n are the values used to find the maximum

Example: To find the Minimum of the given values in the sample sheet (How much spent (column E)) Steps

1. Enter the data as shown in the above sample spread sheet.
2. In D10 location types the text MIN.
3. In E10 type =MIN(E2:E8) and press enter key.
4. The result displayed in E10 is 350

Screen shot:


Book1 - Microsoft Excel
Home Insert Page Layout Formulas Data Review View
Cut Copy Format Painter Paste Clipboard
Calibri 11 Font
Wrap Text Merge & Center Alignment
General Number
Conditional Formatting Format as Table Cell Styles Styles

E10 =MIN(E2:E8)

	A	B	C	D	E
1	S.no	Data	What is it that I spent on?	Category	How much?
2	1	1/1/2015	Rent	rent	2700
3	2	2/1/2015	maid salary	utils	350
4	3	5/1/2015	Internet Bill	utils	1300
5	4	6/1/2015	Electricity	utils	800
6	5	23/1/2015	Groceries	Food	1450
7	6	24/1/2015	Milk	Food	1610
8	7	30/1/2015	Petrol in car	Transport	500
9					
10				MIN	350

2.2.6. Data handling – *If* function to check a condition

Explanation

To check a particular conditions using if function

Formula in the spread sheet:

IF(Logical_test,[value_if_true],[value_if_false])

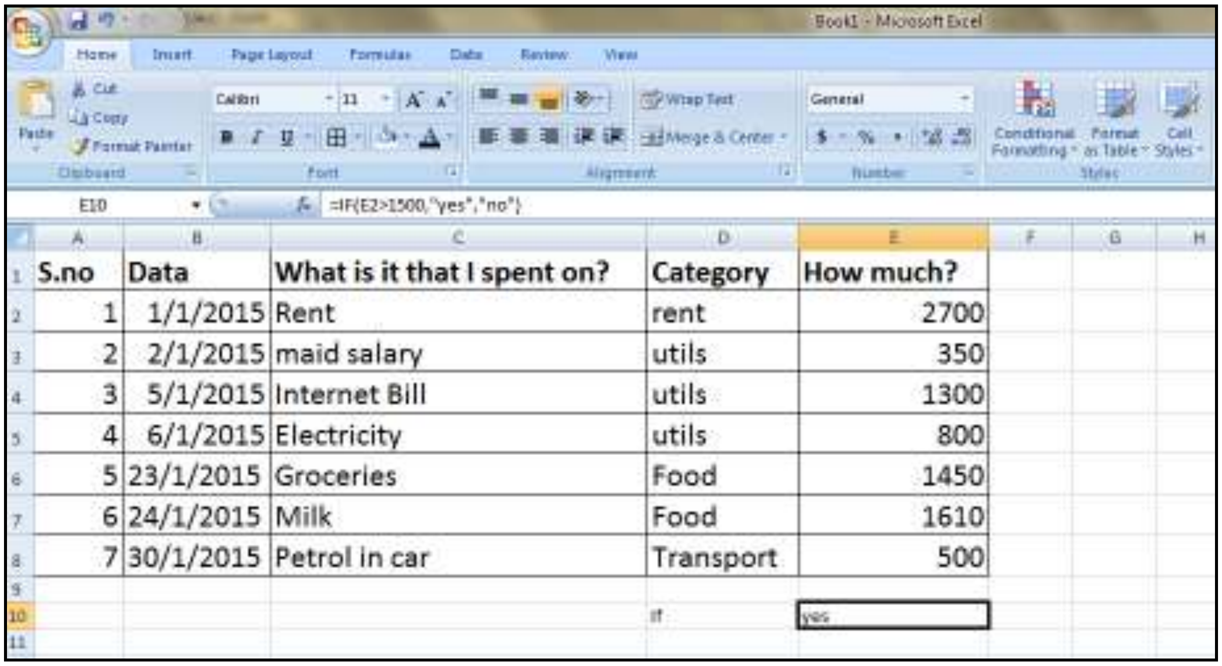
It displays *value_if_true* if logical_test value is true otherwise *value_if_false* will be displayed

Example: To apply if condition for a given value in the sample sheet (How much spent (column E))

Steps

1. Enter the data as shown in the above sample spread sheet.
2. In D10 location types the text If.
3. In E10 type =IF(E2>1500,"yes","no")) and press enter key.
4. The result displayed in E10 is yes

Screen shot:



E10 =IF(E2>1500,"yes","no")

S.no	Data	What is it that I spent on?	Category	How much?
1	1/1/2015	Rent	rent	2700
2	2/1/2015	maid salary	utils	350
3	5/1/2015	Internet Bill	utils	1300
4	6/1/2015	Electricity	utils	800
5	23/1/2015	Groceries	Food	1450
6	24/1/2015	Milk	Food	1610
7	30/1/2015	Petrol in car	Transport	500
			if	yes

2.2.7. Data handling – *COUNTIF* function to count based on a condition

Explanation

To count a particular number based on a criteria using countif function

Formula in the spread sheet: COUNTIF(range,criteria)

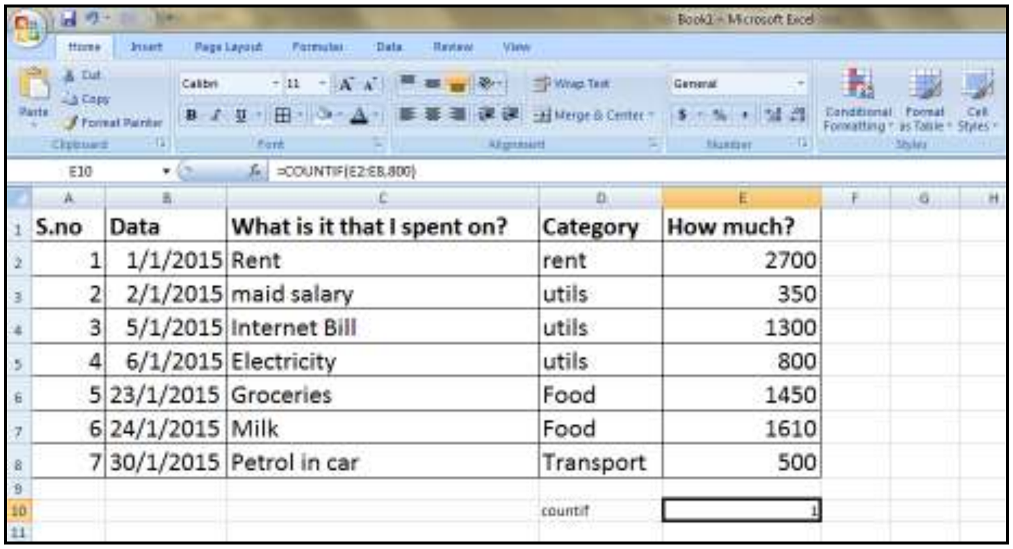
Will display the count based on a criteria

Example: To apply countif condition for a value in the sample sheet (How much spent (column E))

Steps

1. Enter the data as shown in the above sample spread sheet.
2. In D10 location types the text countIf.
3. In E10 type =COUNTIF(E2:E8,800) and press enter key.
4. The result displayed in E10 is 1

Screen shot:



E10 =COUNTIF(E2:E8,800)

S.no	Data	What is it that I spent on?	Category	How much?
1	1/1/2015	Rent	rent	2700
2	2/1/2015	maid salary	utils	350
3	5/1/2015	Internet Bill	utils	1300
4	6/1/2015	Electricity	utils	800
5	23/1/2015	Groceries	Food	1450
6	24/1/2015	Milk	Food	1610
7	30/1/2015	Petrol in car	Transport	500
			countif	1

2.2.8. Data handling – *SUMIF* function

Explanation

To sum a set of values based on a a particular criteria using SUMIF function

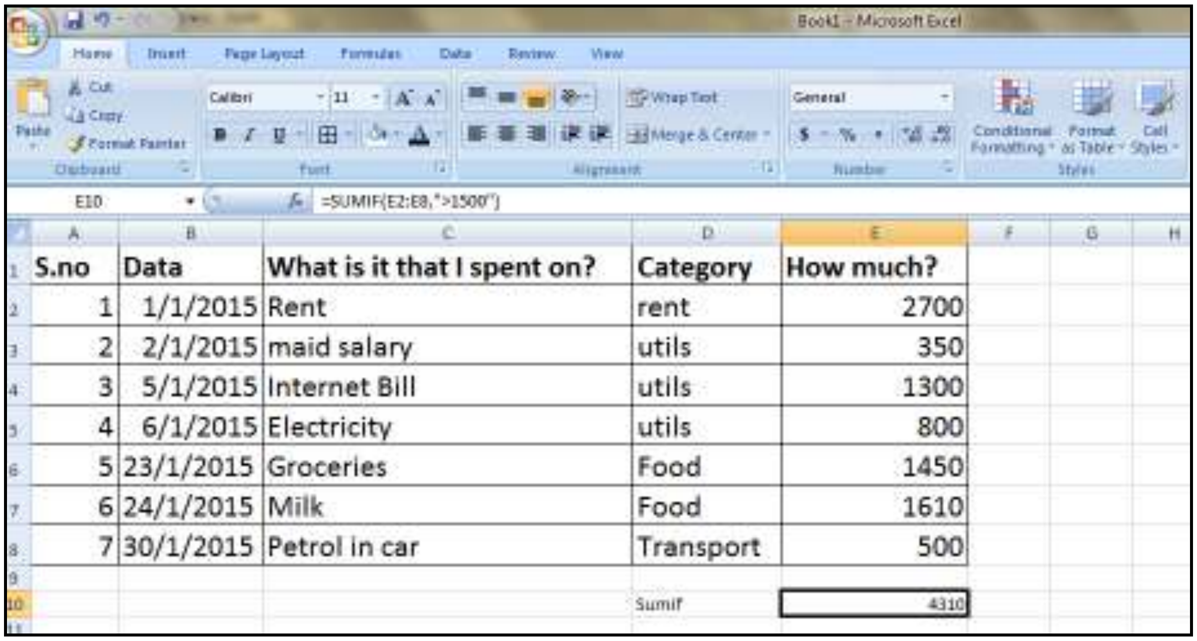
Formula in the spread sheet: SUMIF(range, criteria,[sum_range]))
Will display the sum of values in the given range based on a criteria

Example: To apply sumif condition for a set of values in the sample sheet (How much spent (column E))

Steps

1. Enter the data as shown in the above sample spread sheet.
2. In D10 location types the text Sumif.
3. In E10 type =SUMIF(E2:E8,">1500") and press enter key.
4. The result displayed in E10 is 4310

Screen shot:



E10 =SUMIF(E2:E8,">1500")

S.no	Data	What is it that I spent on?	Category	How much?
1	1/1/2015	Rent	rent	2700
2	2/1/2015	maid salary	utils	350
3	5/1/2015	Internet Bill	utils	1300
4	6/1/2015	Electricity	utils	800
5	23/1/2015	Groceries	Food	1450
6	24/1/2015	Milk	Food	1610
7	30/1/2015	Petrol in car	Transport	500
			Sumif	4310

2.2.9. Data handling – user defined formulae

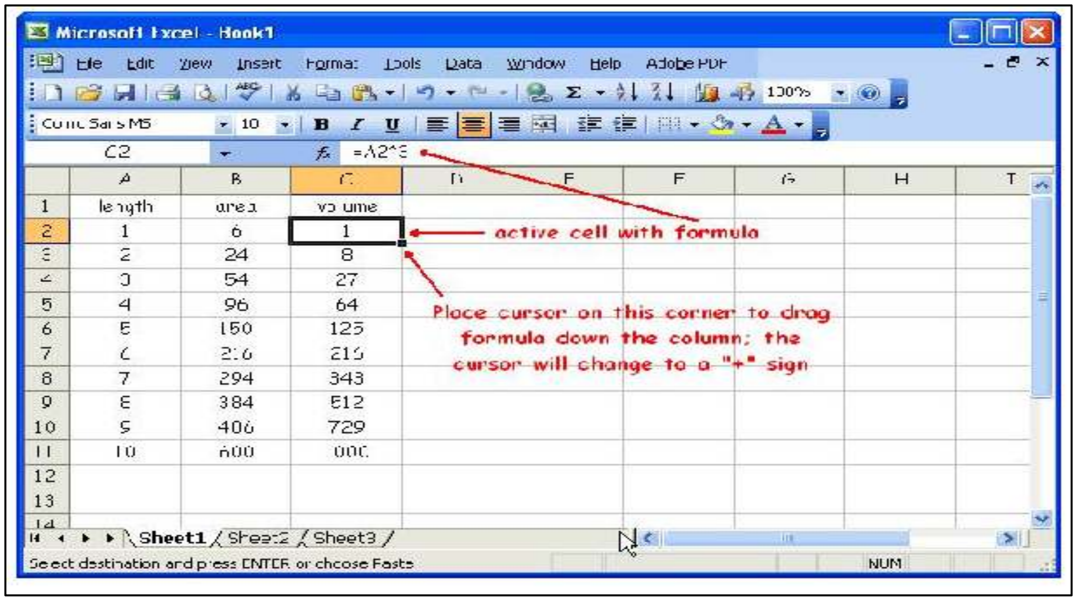
Let's calculate the surface area and volume of cubes of varying edge length as outlined below.

Example

1. Enter the data below into column A: Label length in Al and then start the data in A2: 1, 2, 3, ... to 10

The second task is to transform or manipulate data.

2. In B1 type "Area" and in B2 place the formula =6*A2^2 and then press the enter key. All calculations with a formula must start with an "=" in the cell. This calculates the surface area of the cube. Grab the lower right hand corner of the cell and drag down the column. This will copy the formula to the covered cells. See the screen shot given below.



C2 =A2^3

	A	B	C
1	length	area	volume
2	1	6	1
3	2	24	8
4	3	54	27
5	4	96	64
6	5	150	125
7	6	216	216
8	7	294	343
9	8	384	512
10	9	486	729
11	10	600	1000

active cell with formula

Place cursor on this corner to drag formula down the column; the cursor will change to a "+" sign

Select destination and press ENTER or choose Paste

3. In C1 type "Volume" and in C2 place the formula =A2^3 and then press the enter key. This calculates the volume of the cube. Grab the lower right hand corner of the cell. When the cursor changes form to +, drag the formula down the column.

4. The data and two calculated quantities, surface area and volume, should look like the screen shot above when you are done.

If you right click on the column heading (A, B, C...) you can format the cells for the number of decimal places or significant figures needed. This is very important when considering data generated by experimental measurements in science.

2.3. Exercises

LAQ2.1. You have a monthly income of Rs 1100. Your monthly outgoings are Rent, Rs 500, Food, Rs 300, Electricity, Rs 40, Phone, Rs 60, and Cable TV, Rs 30. Make out a clear worksheet with the Monthly Income, the Monthly Expenses listed and summed, the remainder (what's left over each month) calculated, and the amount left over per day (assuming 30 days in a month)

LAQ2.2. A home maker wants to maintain the monthly expenses using an Excel sheet.

Date	Details for the expenditure	Amount spent
01-05-2015	Rent	5000
01-05-2015	Servant maid	2000
02-05-2015	Electricity Bill	1000
03-05-2015	Grocery	5000
	Total	13000

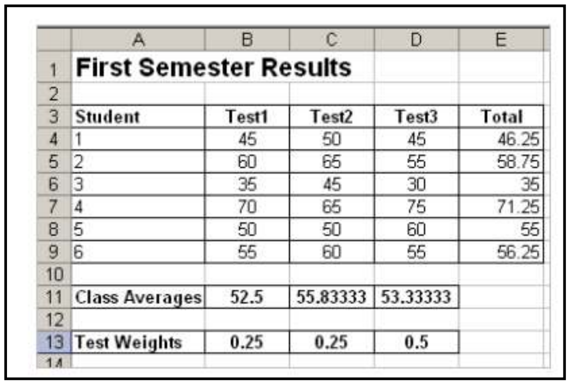
Note: Create a sub total of the amount spent for every 10 days and calculates the average amount spent for the month of May.

LAQ2.3. A faculty handling "*Computational thinking*" subject wants to publish internal marks for the students using an Excel Sheet.

S.no	Roll No	Name	C1 (25)	C2 (25)	C3(40)	C4(10)	Internals(100)
1	001	Rama	12	12	40	10	74
2	002	Sita	12	12	30	10	64
3	003	Tom	10	10	40	10	70

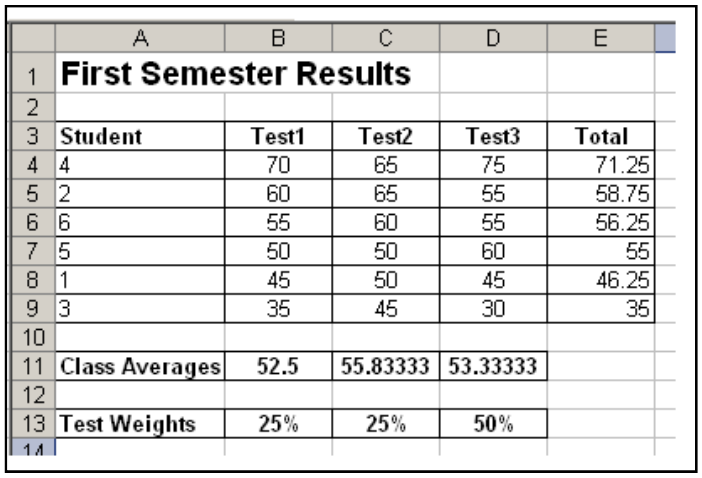
Calculate minimum internal mark scored, maximum internal mark scored, sum of the each component marks for a student, sum of all the internal marks scored by the students, average mark of the class, standard deviation, and median. Count the number of students who scored less than 50, 51 – 60, 61 – 70, 71 – 80, 81 – 90, 91 -100. Finally sort the details in descending order based on the internal marks.

LBQ2.4. Create the worksheet below. Type in the labels, the student numbers, the test results (which are out of 100), and the test weights. The total marks and the class averages for each test must be calculated. The class average for test is simply the average of the marks obtained in that test. While calculating the total marks - you must use the test weights. The tests have the following weights: 25% of the total marks for test1 & test2 and 50% of the total marks for test 3. Changing the weights in the cells should change the corresponding weighted values.



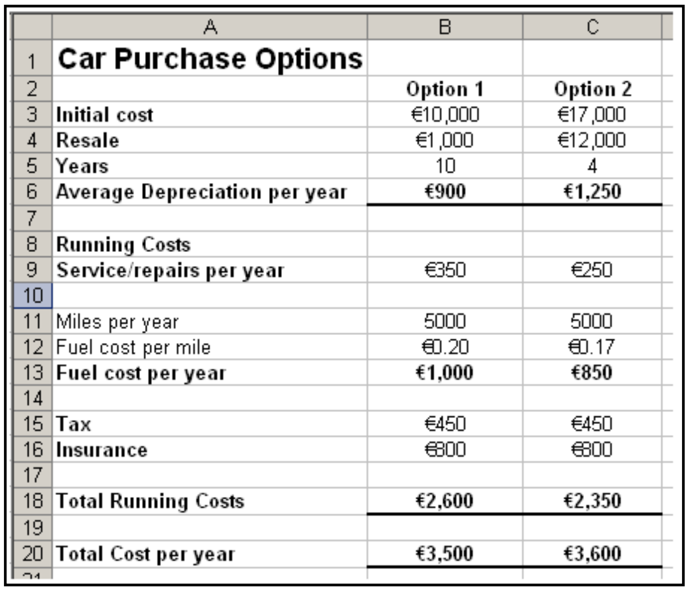
	A	B	C	D	E
1	First Semester Results				
2					
3	Student	Test1	Test2	Test3	Total
4	1	45	50	45	46.25
5	2	60	65	55	58.75
6	3	35	45	30	35
7	4	70	65	75	71.25
8	5	50	50	60	55
9	6	55	60	55	56.25
10					
11	Class Averages	52.5	55.83333	53.33333	
12					
13	Test Weights	0.25	0.25	0.5	

Now format the test weights so that they are shown as percentages. Simply select the cells, choose FORMAT|CELLS, and format them as "Percentage". Select the cells from A4 to E9 and choose DATA|SORT. Select that you wish to sort according to the 'Total' column and that you want the results sorted in Descending order. You should end up with



	A	B	C	D	E
1	First Semester Results				
2					
3	Student	Test1	Test2	Test3	Total
4	4	70	65	75	71.25
5	2	60	65	55	58.75
6	6	55	60	55	56.25
7	5	50	50	60	55
8	1	45	50	45	46.25
9	3	35	45	30	35
10					
11	Class Averages	52.5	55.83333	53.33333	
12					
13	Test Weights	25%	25%	50%	

LAQ2.5. You are looking to buy a car. You are considering two options: to buy a second hand car and keep it for 10 years, or to buy a new car and keep it for 4. The details of the costs of each option are given below. The calculated values are shown in Bold. The depreciation per year is simply the difference between the purchase price and the resale price divided by the number of years. The total running costs per year is the sum of the service/repair costs, the fuel costs, the tax, and the insurance. The total cost per year is the sum of the depreciation and the running costs. Note that the underlining can be done with one of the options under the Borders toolbar button.



	A	B	C
1	**Car Purchase Options**		
2		**Option 1**	**Option 2**
3	**Initial cost**	€10,000	€17,000
4	**Resale**	€1,000	€12,000
5	**Years**	10	4
6	**Average Depreciation per year**	**€900**	**€1,250**
7			
8	**Running Costs**		
9	**Service/repairs per year**	€350	€250
10			
11	Miles per year	5000	5000
12	Fuel cost per mile	€0.20	€0.17
13	**Fuel cost per year**	**€1,000**	**€850**
14			
15	**Tax**	€450	€450
16	**Insurance**	€800	€800
17			
18	**Total Running Costs**	**€2,600**	**€2,350**
19			
20	**Total Cost per year**	**€3,500**	**€3,600**

A third option is to purchase the new car for €17,000 and run it for 14 years after which it will be worth €1000. The average service/repairs cost per year will be about €320. The average fuel cost per mile over the 14 years will be about €0.18. The tax and insurance will be the same. Calculate the Total cost per year for Option 3.

LBQ2.6. A car was accelerated up to 120 kmph and the speed was measured at two second intervals. The results are,

Time (s)	Speed (kmph)
0	0
2	35
4	60
6	78
8	95
10	110
12	120

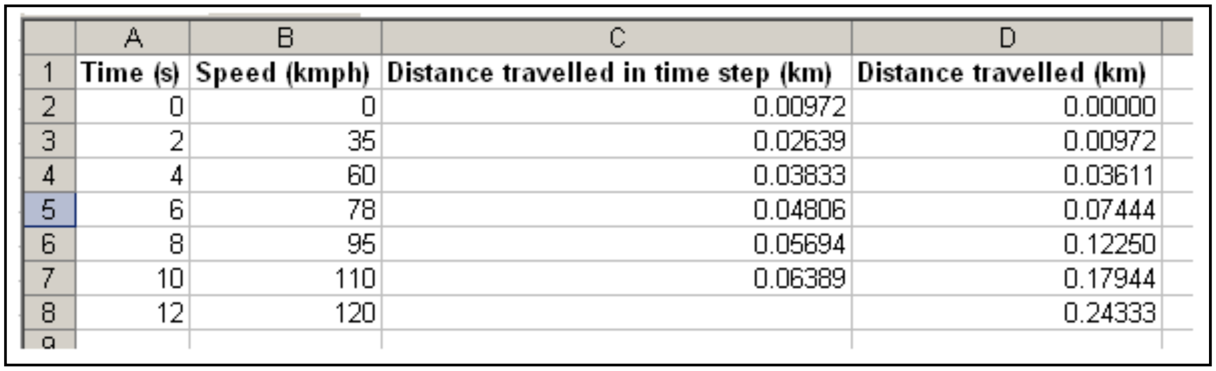
You will use this data to calculate the acceleration of the car and how it changes over time. Acceleration is the rate of change of velocity (or speed if you're not considering the direction of the motion). You can estimate it from the values in the table. The speed changed from 0 kmph to 35 kmph in the first 2 seconds so we can assume that the acceleration at time 0 s was approximately (35-0)/2 kmph per second. Use Excel to calculate the acceleration at the other points in time – first enter the values above and then add a new column for the acceleration. You do not have to calculate the acceleration at time 12 seconds. Graph the acceleration values against time. Label the graph correctly.

LBQ2.7. Given the speed data from the table in Exercise 4 above, calculate how far the car has travelled at each point in time. How will you do this?

If a car is going at 80 kmph for 2 hours, how far has it travelled? – 80 kmph * 2 hours = 160 km.

If a car starts off at 40 kmph and ends up at 80kmph after 2 hours, how far has it travelled in the two hours? The answer is that we don’t know. A reasonable way of estimating the answer, however, is to take the average speed ((40 + 80) / 2 = 60 kmph) and multiply it by 2 hours.

In this example you can estimate how far the car has travelled in each two second interval. Note that there are 3600 seconds in an hour. Add these values up to find out how far it has travelled at any point in time. The solution is shown below. Plot the distance travelled against time.



	A	B	C	D
1	Time (s)	Speed (kmph)	Distance travelled in time step (km)	Distance travelled (km)
2	0	0	0.00972	0.00000
3	2	35	0.02639	0.00972
4	4	60	0.03833	0.03611
5	6	78	0.04806	0.07444
6	8	95	0.05694	0.12250
7	10	110	0.06389	0.17944
8	12	120		0.24333
9				

3. LOGIC – BOOLEAN AND SIMPLE PROPOSITIONAL LOGIC

3.1. True/False Propositions – Checking Equality

Explanation

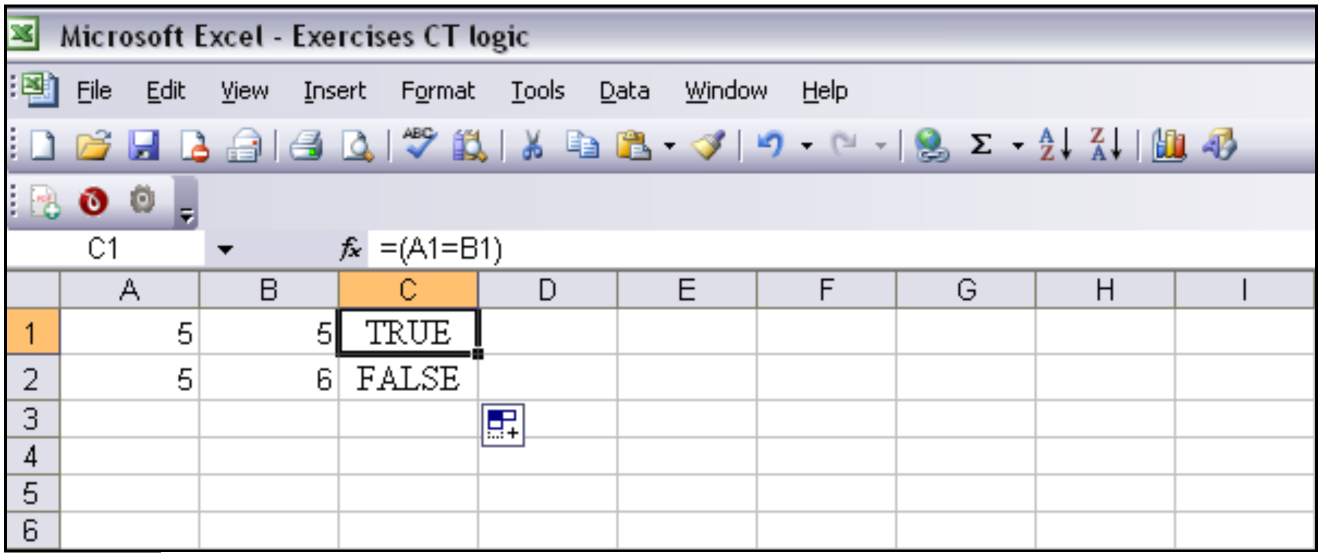
It checks if two quantities are *equal*. If the two quantities are equal, True is displayed. If the quantities are not equal, displays False

Formula in spreadsheet: Value1 = Value2
where Value1, Value2 are values contained in two cells.

Example: To check whether the number 5 is equal to the number 5.

Steps:
1. Open a spread sheet
2. Type the number 5 in the cell in row 1 under column A
3. Type the number 5 in the cell in row 1 under column B
4. Click the cell in row 1 under column C and type =(A1=B1) and press Enter key
5. The result displayed in the cell C1 is TRUE

Screen Shot:



C1 =(A1=B1)

	A	B	C	D	E	F	G	H	I
1	5	5	TRUE						
2	5	6	FALSE						
3									
4									
5									
6									

3.2. True/False Propositions – Checking Inequality

Explanation

It checks if two quantities are *not equal*. If the two quantities are not equal, True is displayed. If the quantities not equal, displays False

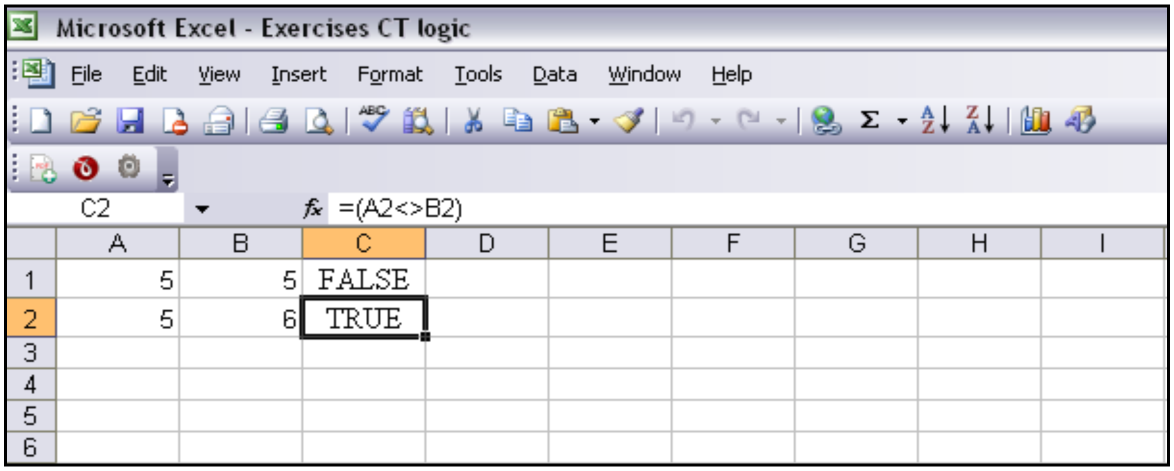
Formula in spreadsheet: Value1 <> Value2

where Value1, Value2 are values contained in two cells.

Example: To check the inequality of two numbers 5 and 6.

Steps:
1. Open a spread sheet
2. Type the number 5 in the cell in row 2 under column A
3. Type the number 6 in the cell in row 2 under column B
4. Click the cell in row 2 under column C and type =(A1<>B1) and press Enter key
5. The result displayed in the cell C2 is TRUE

Screen Shot:


Microsoft Excel - Exercises CT logic
File Edit View Insert Format Tools Data Window Help

C2 =(A2<>B2)

	A	B	C	D	E	F	G	H	I
1	5	5	FALSE						
2	5	6	TRUE						
3									
4									
5									
6									

3.3. NOT – Negation

Explanation

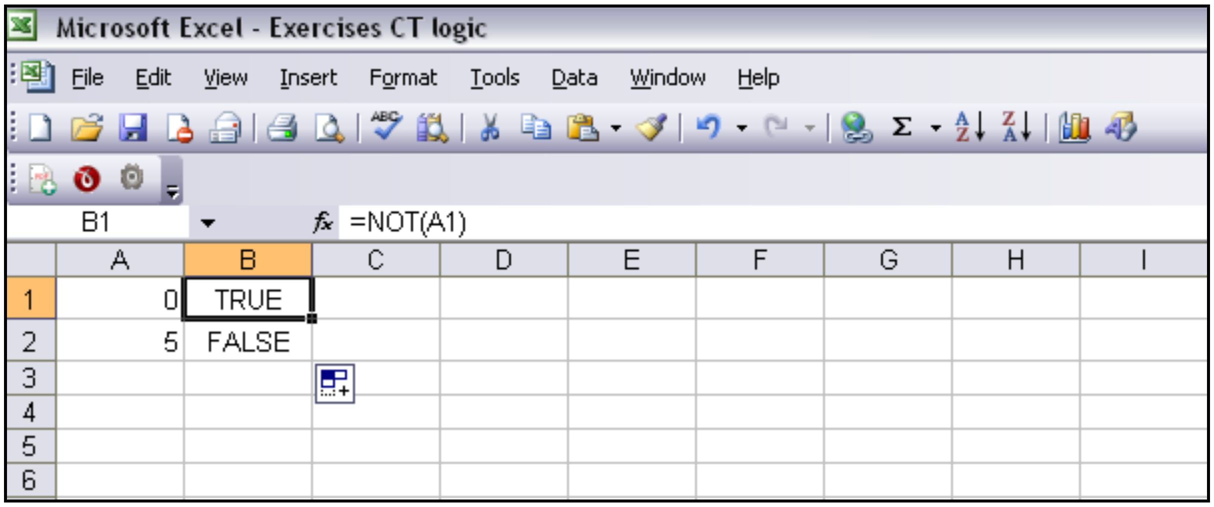
Converts a non-zero (TRUE) value to zero (FALSE) and a zero (FALSE) value to non-zero (TRUE) value

Formula in spreadsheet: =NOT(cellvalue)

Example: To convert a non-zero number to TRUE and vice versa.

Steps:
1. Open a spread sheet
2. Type the number 0 in the cell in row 1 under column A
3. Type the number 5 in the cell in row 2 under column A
4. Click the cell in row 1 under column B and type =NOT(A1) and press Enter key
5. The result displayed in the cell B2 is TRUE
6. Click on B1 cell and a dark rectangle highlights the cell
7. Click on the small square at the right bottom corner of the rectangle
8. Click and hold your left mouse button and drag the above square till cell B2
9. FALSE is displayed in B2.

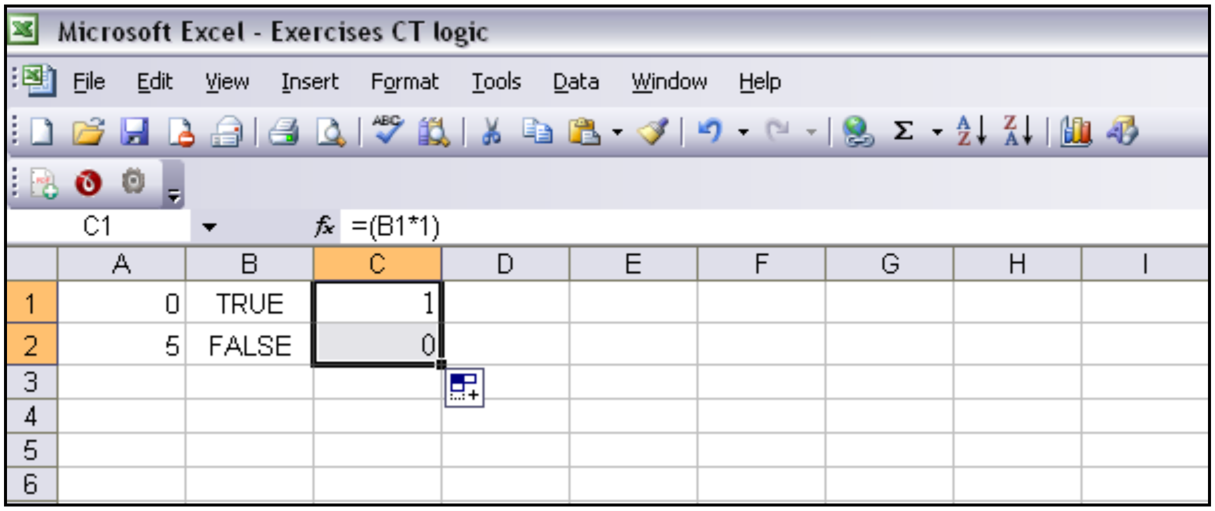
Screen Shots:


Microsoft Excel - Exercises CT logic
File Edit View Insert Format Tools Data Window Help

B1 =NOT(A1)

	A	B	C
1	0	TRUE	
2	5	FALSE	

Note: The values TRUE and FALSE can be converted into Boolean values 0 and 1 by multiplying the Formula by 1 i.e., =NOT(cellvalue)*1 would display the result as 1 and 0 as shown in the below screen shot:


Microsoft Excel - Exercises CT logic
File Edit View Insert Format Tools Data Window Help

C1 =(B1*1)

	A	B	C
1	0	TRUE	1
2	5	FALSE	0

3.4. AND – Conjunction

Explanation

The result is TRUE only if all the cell values are non-zero (TRUE)

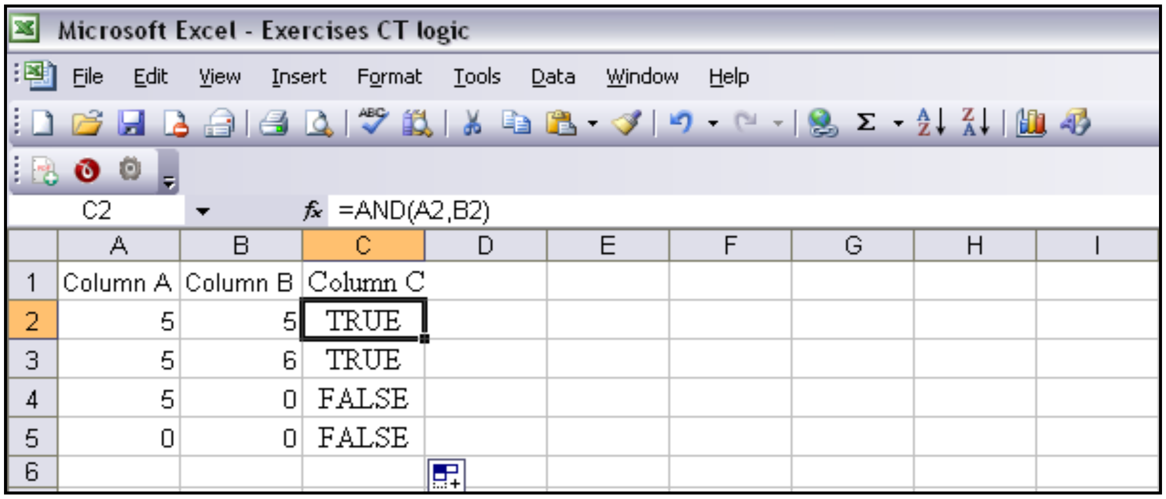
Formula in spreadsheet: =and(cellvalue1,cellvalue2,…cellvalue n)

Example: To check the conjunction between two columns for different combinations of cell values

Steps:
1. Open a spread sheet
2. Type the numbers under column A (from A2) as given in the below screen shot
3. Type the numbers under column B (from B2) as given in the below screen shot

4. Under column C2 write the formula =AND(A2,B2)
5. The result displayed in the cell C2 is TRUE
6. Click on C2 cell and a dark rectangle highlights the cell
7. Click on the small square at the right bottom corner of the rectangle
8. Click and hold your left mouse button and drag the above square till cell C5
9. All the cells are filled with appropriate conjunction results

Screen Shot:



=AND(A2,B2)

	A	B	C	D	E	F	G	H	I
1	Column A	Column B	Column C						
2	5	5	TRUE						
3	5	6	TRUE						
4	5	0	FALSE						
5	0	0	FALSE						
6									

3.5. OR – Disjunction

Explanation

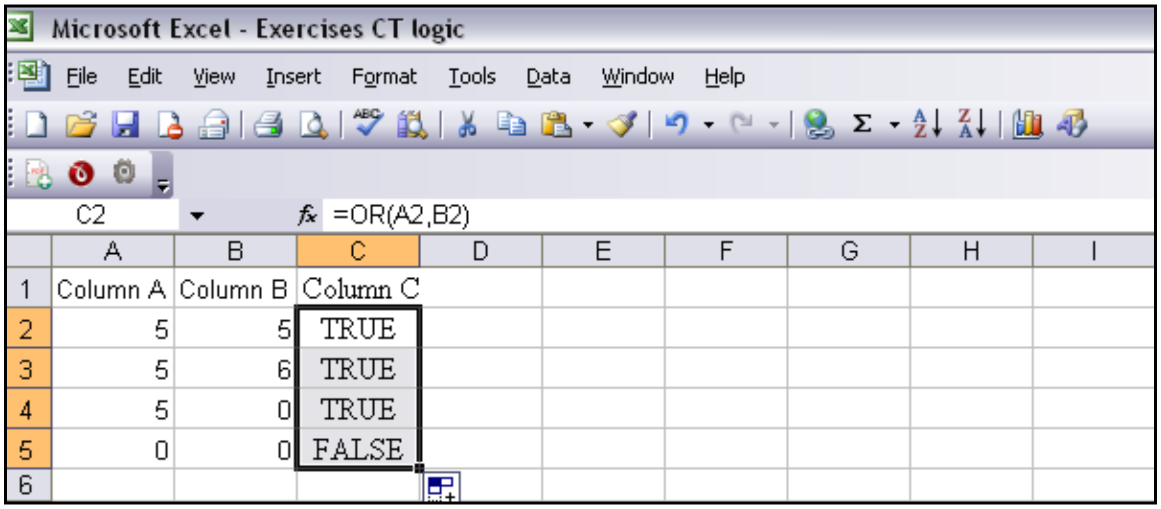
The result is TRUE even if one of the values are non-zero (TRUE)

Formula in spreadsheet: =or(cellvalue1,cellvalue2,…cellvalue n)

Example: To check the disjunction between two columns for different combinations of cell values

Steps:
1. Open a spread sheet
2. Type the numbers under column A (from A2) as given in the below screen shot
3. Type the numbers under column B (from B2) as given in the below screen shot
4. Under column C2 write the formula =OR(A2,B2)
5. The result displayed in the cell C2 is TRUE
6. Click on C2 cell and a dark rectangle highlights the cell
7. Click on the small square at the right bottom corner of the rectangle
8. Click and hold your left mouse button and drag the above square till cell C5
9. All the cells are filled with appropriate disjunction results

Screen Shot:


Microsoft Excel - Exercises CT logic
File Edit View Insert Format Tools Data Window Help
C2
=OR(A2,B2)

	A	B	C	D
1	Column A	Column B	Column C	
2	5	5	TRUE	
3	5	6	TRUE	
4	5	0	TRUE	
5	0	0	FALSE	
6				

3.6. Exercises

LAQ3.1. The electoral commission of India is in the process of consolidating and updating the names of people in a particular area. School teachers are going around in each area collecting the names of people in each house one after the other and that list is checked with the existing list in the electoral office to find if both the lists match or else the list are updated with the names collected by the teachers. Devise a mechanism to compare the two lists in case tenants of certain houses have left and new people have moved in.

LAQ3.2. Excel University provides Computers to its faculty members and the assets team performs a periodical stock taking to check if the same machine is still used by the faculty or it has been replaced by some other machine for some reason by the systems team. A machine could be changed if it is upgraded or it is repaired or if it is replaced by some other. A copy of the list resides in the assets office and it is checked with the list noted by the stock taker and those machines which do not match with the old list need to be updated with the new details. Prepare a data sheet containing cabin numbers (numbered F1, F2, F20) machine models (Excelsys1100,…) and verify with another set similar data obtained by the stock taking operation and indicate the matching and non-matching entries in the sheet which needs updation.

LBQ3.3. Solve the problem Q3.1 in previous section using inequality operator.

LBQ3.4. Solve the problem Q3.2 in previous section using inequality operator.

LAQ3.5. An Un-interrupted Power Supply (UPS) works on the fact that the presence of electricity turns off the UPS and the absence of electricity turns on the UPS. Model this operation of an UPS using logic.

LBQ3.6. Mr. Anthony is very much concerned about the depletion of energy reserves and has installed in his home both geyser and solar water heater in his bathroom. When the day is hot and the solar heater is heating, the geyser remains off and when the day is cloudy, the geyser is on. Model this scenario using NOT logic.

LBQ3.7. Michael works as a *marketing executive* and spends his whole day travelling around getting orders for his company. Sometimes the trips are short and often times he has to travel large distances on his two wheeler. He decides his outfit based on the climate so that he is comfortable when he is at work. The decision is made as follows: If the day is cold and also it is snowing, he wears a parka. If it is cold and not snowing, he wears a jacket. If the day is not cold, He wears formal outfit. Design an automatic decision making system to help Michael to choose his outfit, given a list of day type combinations (cold and snowing).

LBQ3.8. Simson works as a boiler controller in *'Powerful Boiler Industries'* as a boiler operator for power generation. There are certain guidelines for the operation of the boiler to work in standard operating conditions. Any kind of abnormal function is notified by a temperature indicator which measures the temperature in the boiler and sounds an appropriate alarm. If the temperature falls below 140 F for 5 minutes and the thermostat is not on to turn on the heating burner then the 'under heat alarm' signals. If the temperature is between 140 F and 200 F and the thermostat calls for burner off, then the 'normal alarm' signals. If the temperature is greater than 200 F and the time is greater than 10 minutes and the thermostat is not calling for burner off, 'over heat alarm' signals. Model the outputs for this scenario using sample temperatures, time and burner inputs.

LAQ3.9. A physical electrical circuit consists of two voltages V_a and V_b which can take high values or low values. High voltage signifies that current is flowing; low voltage signifies that there is no current. When any one of the voltages is high it signifies that current is flowing and hence caution must be exercised. Model this electrical problem using propositional logic.

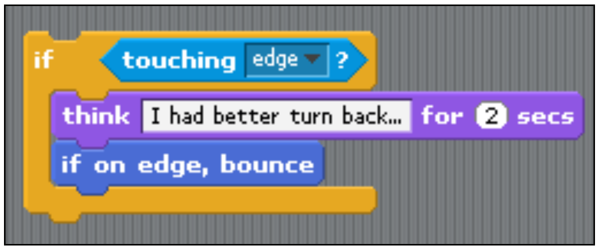
LBQ3.10. A person is eligible for obtaining *Gas subsidy* if he has submitted his *Aadhar card* or has a *bank account*. '*Efficient Gas Agency'* is in the process of verifying whether its consumers have submitted either of the two documents and come up with a list whether a customer is eligible for subsidy or not. The data sheet consists of Consumer Number, Consumer Name, Aadhar card Submission status and Bank account number submission status. Design a strategy to determine the list of consumers of Efficient Gas Agency who are eligible for gas subsidy and those who are not.

4. SELECTION AND REPETITION TECHNIQUES

4.1. Conditional statements

Conditional statements ask questions about the program state to choose from a set of different sequences of commands. In Scratch for example, you can determine whether you are at the edge of the stage with the **if touching edge** control block. The following example (implemented within a forever loop on the shark in the conditional project) causes the sprite to think each time it reaches the edge of the stage before bouncing off again:

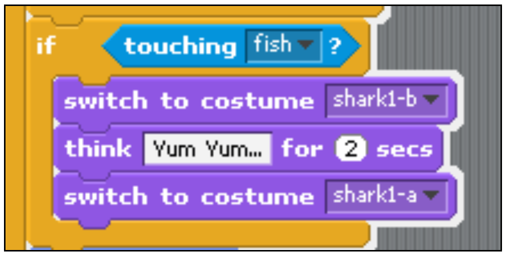
Example -1



```
if touching edge ?
    think I had better turn back... for 2 secs
    if on edge, bounce
```

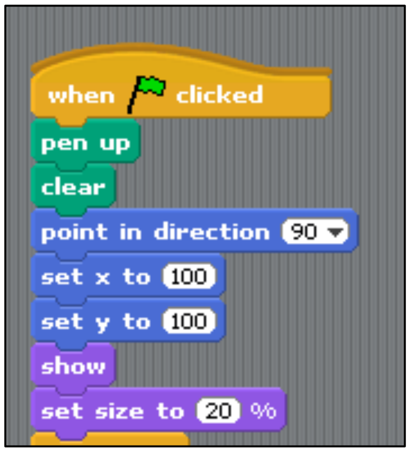
Example -2

The example conditional project implements this on the fish sprite with a corresponding change of costume on the shark so it looks like the shark eats the fish when they touch:



```
if touching fish ?
    switch to costume shark1-b
    think Yum Yum... for 2 secs
    switch to costume shark1-a
```

Pressing the green flag will need to restore the fish sprite back to life by showing the sprite, setting the initial position and size and starting the forever loop:

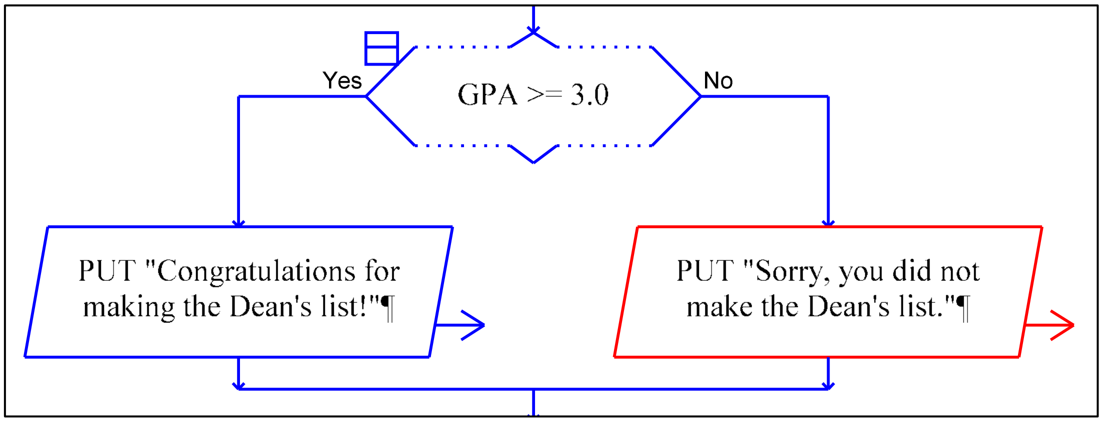


```
when clicked
pen up
clear
point in direction 90
set x to 100
set y to 100
show
set size to 20 %
```

4.2. Control Flow

Algorithms use control flow to make decisions about which order to do things. They can repeat actions or start new actions based on new information. Computer programs use sequence, selection and iteration to control the flow of the program. These elements allow the program to make choices, change direction or repeat actions.

Example: To find student with GPA greater than or equal to 3.0

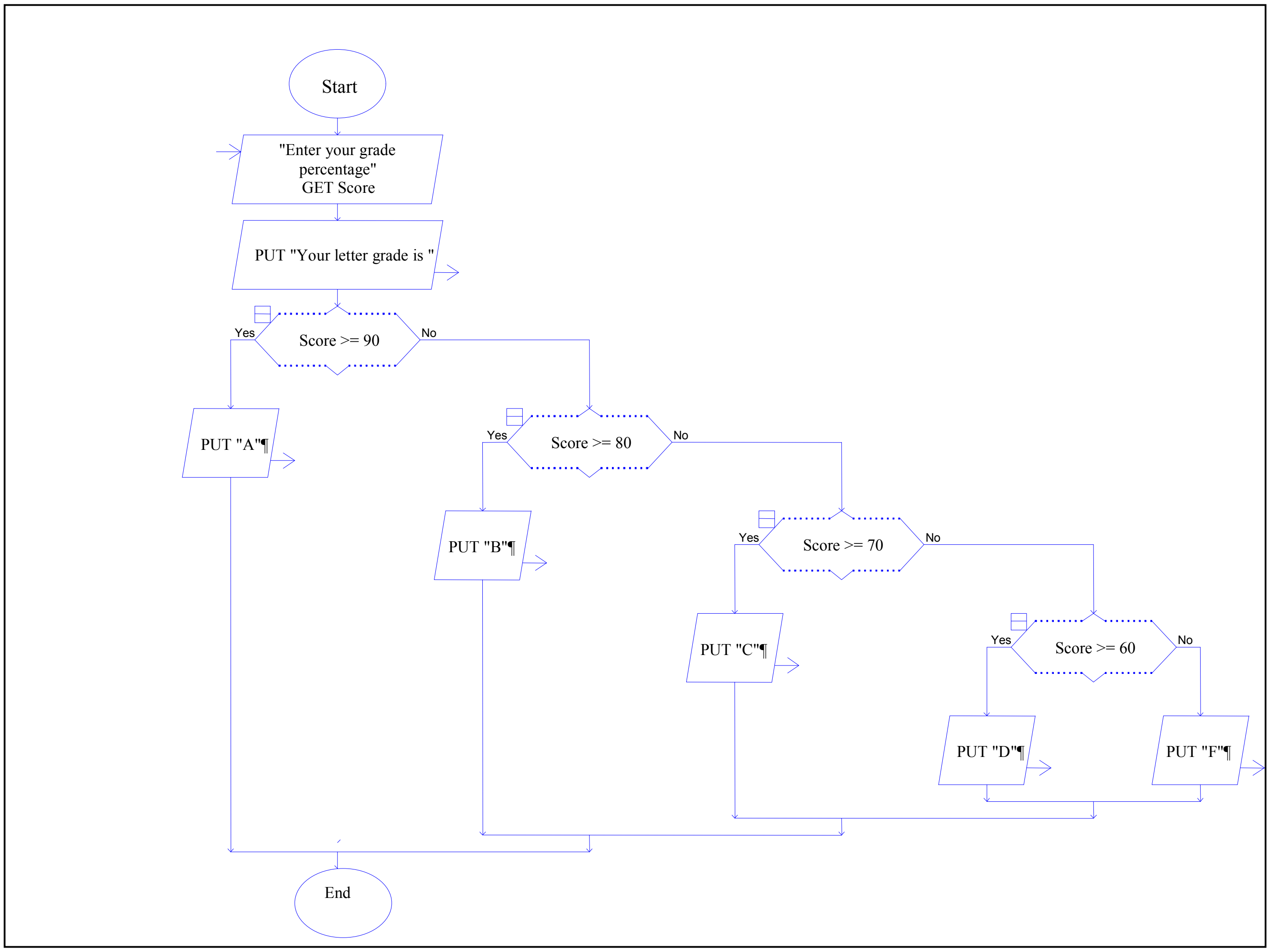

Yes
GPA >= 3.0
No
PUT "Congratulations for making the Dean's list!"¶
PUT "Sorry, you did not make the Dean's list."¶

4.3. Nested Control Flow

A single selection-control statement can make a choice between one or two choices. If you need to make a decision that involves more than two choices, you need to have multiple selection control statements.

For example, if you are assigning a letter grade (A, B, C, D, or F) based on a numeric score, you need to select between five choices, as shown below. This is sometimes referred to as "cascading selection control," as in water cascading over a series of waterfalls.

Example: To get the marks and display the grade

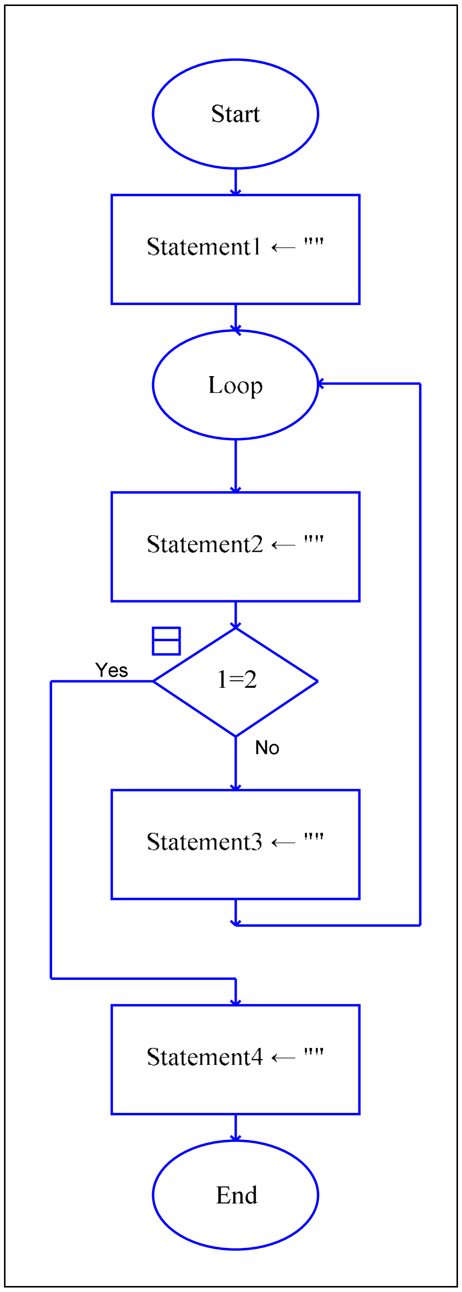


4.4. Repetitive Control Structures -1

A Loop (iteration) control statement allows you to repeat one or more statements until some condition becomes true. This type of control statement is what makes computers so valuable. A computer can repeatedly execute the same instructions over-and-over again without getting bored with the repetition.

One ellipse and one diamond symbol is used to represent a loop in RAPTOR. The number of times that the loop is executed is controlled by the decision expression that is entered into the diamond symbol. During execution, when the diamond symbol is executed, if the decision expression evaluates to "no," then the "no" branch is taken, which always leads back to the Loop statement and repetition. The statements to be repeated can be placed above or below the decision diamond. To understand exactly how a loop statement works, study the example RAPTOR program to the right and notice the follow things about this program:

- Statement 1 is executed exactly once before the loop (repetition) begins.
- Statement 2 will always be executed at least once because it comes before the decision statement.
- If the decision expression evaluates to "yes," then the loop terminates and control is passed to Statement
- If the decision expression evaluates to "no," then control passes to Statement 3 and Statement 3 is executed next. Then control is returned back up to the Loop statement which re-starts the loop.
- Note that Statement 2 is guaranteed to execute at least once. Also note that Statement 3 is possibly never executed.



There are too many possible executions of this example program to list them all, but a few of the possibilities are listed in the following table. Make sure you can fill in fourth column with the correct pattern.

Possibility 1	**Possibility 2**	**Possibility 3**	**Possibility 4**
Statement 1 Statement 2 Decision ("yes") Statement 4	Statement 1 Statement 2 Decision ("no") Statement 3 Statement 2 Decision ("yes") Statement 4	Statement 1 Statement 2 Decision ("no") Statement 3 Statement 2 Decision ("no") Statement 3 Statement 2 Decision ("yes") Statement 4	(do you see the pattern?)

In the RAPTOR example above, "Statement2" could be removed, which means that the first statement in the loop would be the "Decision" statement. Or "Statement2" could be a block of multiple statements. In either case the loop executes in the same way. Similarly, "Statement3" could be deleted or be replaced by multiple statements. In addition, any of the statements above or below the "Decision" statement could be another loop statement! If a loop statement occurs inside a loop, we called these "nested loops."

It is possible that the "Decision" statement never evaluates to "yes." In such a case you have an "infinite loop" that will never stop repeating. (If this ever happens, you will have to manually stop your program by selecting the "stop" icon in the tool bar.) You should never write statements that create an infinite loop. Therefore, one (or more) of the statements in the loop must change one or more of the variables in the "Decision" statement such that it will eventually evaluate to "yes."

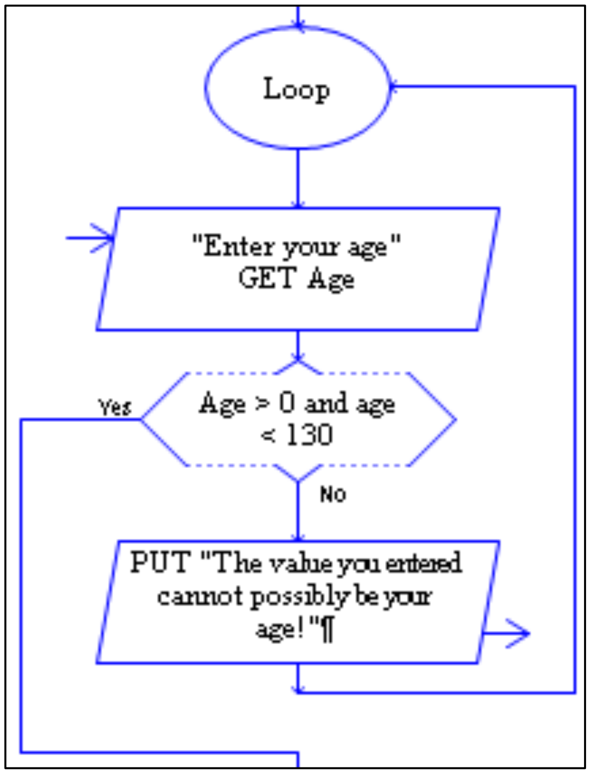
Input Validation Loops

One common use for a loop is to validate user input. If you want the user to input data that meets certain constraints, such as entering a person's age, or entering a number between 1 and 10, then validating the user input will ensure such constraints are met before those values are used elsewhere in your program. Programs that validate user input and perform other error checking at run-time are called robust programs.

A common mistake made by beginning programmers is to validate user input using a selection statement. This can fail to detect bad input data because the user might enter an invalid input on the second attempt. Therefore, you must use a loop to validate user input.

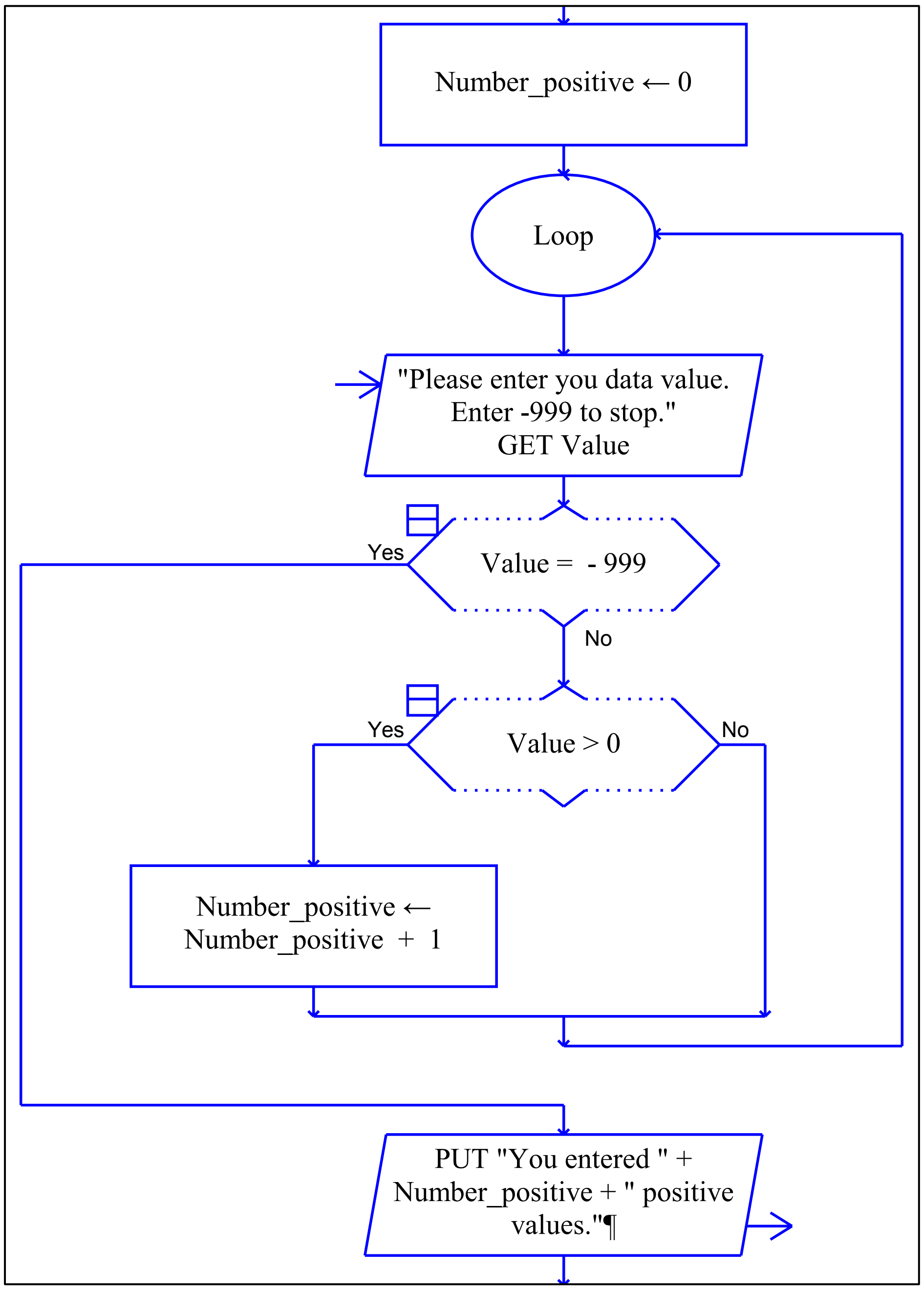
The following example RAPTOR program validates user input. Hopefully you see a pattern. Almost every validation loop that you write will include an input prompt, a decision, and an output error message.

Example: Read age of students and check whether it is valid



4.5. Repetitive Control Structures -2

Example : To count the number of positive integers entered


Number_positive ← 0
Loop
"Please enter you data value. Enter -999 to stop." GET Value
Yes
Value = - 999
No
Yes
Value > 0
No
Number_positive ← Number_positive + 1
PUT "You entered " + Number_positive + " positive values."¶

4.6. Exercises

LAQ4.1. If a user enters a vowel, number or any other character the user gets fixed points in a game. 5 points for vowel, 10 points for number and 0 for any other character. Display the points scored based on the character.

LAQ4.2. User may type uppercase or lowercase letters. When a college is processing admission forms it becomes difficult to process lowercase and uppercase letters. When a user types a character if it is lowercase case convert it to uppercase.

LBQ4.3. A game is devised for learning the nature of roots of a quadratic equation based on the coefficients entered. Based on the coefficients, compute discriminant (b^2 -4ac) and check if the equation has real, imaginary or equal roots. If it is equal roots no points are rewarded. If real roots 20 points are awarded and if it is imaginary 10 points are awarded. Compute the points for a user.

LAQ4.4. Classify a person as Overweight, Normal or Underweight based on the Body Mass Index.

LAQ4.5. Given x-coordinates and y-coordinates find the quadrant in which the point lies. If two points are given find if they form a line.

LBQ4.6. The consistency of a soil can be determined by field tests in which the soil is evaluated in place, or by lab tests on samples that have been carefully handled to avoid remolding. The unconfined compression test is often used as an indication of consistency. In practice, the relative terms soft, medium, stiff, very stiff, and hard are applied to describe consistency. Get the shear strength and classify soil into different consistencies Soft, Medium Stiff, Stiff Very Stiff.

Shear Strength N/m^2	**Consistency**
Less than 24500	Soft
24500-49000	Medium Stiff
49000-98000	Stiff
98000-196000	Very Stiff
Greater than 196000	Hard

LAQ4.7. Suppose data giving the living areas and prices of houses from NewFoundLand are the following. Based on the budget of a customer find the Living area he/she can afford. If it does not fit in his/her budget ask the customer to enter another budget and so on.

Living area	**Price**
2100	14 lakhs
1500	11 lakhs
3100	20 lakhs

LBQ4.8. Voting Machine Company Live bold has discovered a security hole in their machines! In 2008, a few nefarious individuals discovered that certain numeric pass codes would allow access to vote counts for districts all across the country. As it turns out, these special numbers, called powerful numbers, all have a certain property. A powerful number is a positive integer *m*, such that for every prime number *p* dividing *m*, *p*2 also divides *m*. In order to fix this breach, Live bold needs to be able to find all the powerful numbers in a certain range, and they have enlisted you to help.

LAQ4.9. Find the smallest marble from a collection of marbles.

LAQ4.10. A basket ball selection is conducted in RichValley School. Find the tallest student in each class. He/She will be selected as member of the basketball team.

5. RECURSION AND FACTORING TECHNIQUES

5.1. Recursion

Recursion is the process a procedure goes through when one of the steps of the procedure involves invoking the procedure itself. A procedure that goes through recursion is said to be 'recursive'.
One way to think about recursion:

- When a recursive call is made, the method **clones itself**, making new copies of:
    - the code
    - the local variables (with their initial values),
    - the parameters
- Each copy of the code includes a marker indicating the current position. When a recursive call is made, the marker in the old copy of the code is just after the call; the marker in the "cloned" copy is at the beginning of the method.
- When the method returns, **that** clone goes away, but the previous ones are still there, and know what to execute next because their current position in the code was saved (indicated by the marker).

Definition
Base case: The case for which the solution can be stated non-recursively
General (recursive) case: The case for which the solution is expressed in terms of a smaller version of itself
Recursive algorithm A solution that is expressed in terms of (a) smaller instances of itself and (b) a base case

Sample Problems

Problem #1 – Dudeney's Cows

If a cow produces its first she-calf at age two years and after that produces another single she-calf every year, how many she-calves are there after N years, assuming none die?

let N = number of newborn she-calves,
I = number of one-year-old she-calves, and
C = number of cows (2 years old or older)

year	N	I	C
0	1	0	0

Then the next year, the newborn just ages. That is, to find I in the next year, just take N of the previous year, I(k) = N(k-1)
So,

year	N	I	C
0	1	0	0
1	0	1	0

Then the next year, the infant grows into a cow, old enough to have a she-calf. And, just at the end of year 2, the cow is 2 years old, so there's an infant born ... so we end up with

year	N	I	C
0	1	0	0
1	0	1	0
2	1	0	1

Before we try to jump to a formula for those, let's do another couple years. The next year there'll be one more newborn (coming from the 1 cow), and one infant (the newborn ages), and one cow (the cow survives), so we get

year	N	I	C
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	1

Then there'll be two newborns the next year (one from the infant coming of age, and one from the cow), one infant (from the newborn aging), and two cows (one from the infant coming of age, and one from the cow surviving), so we get

year	N	I	C
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	1
4	2	1	2

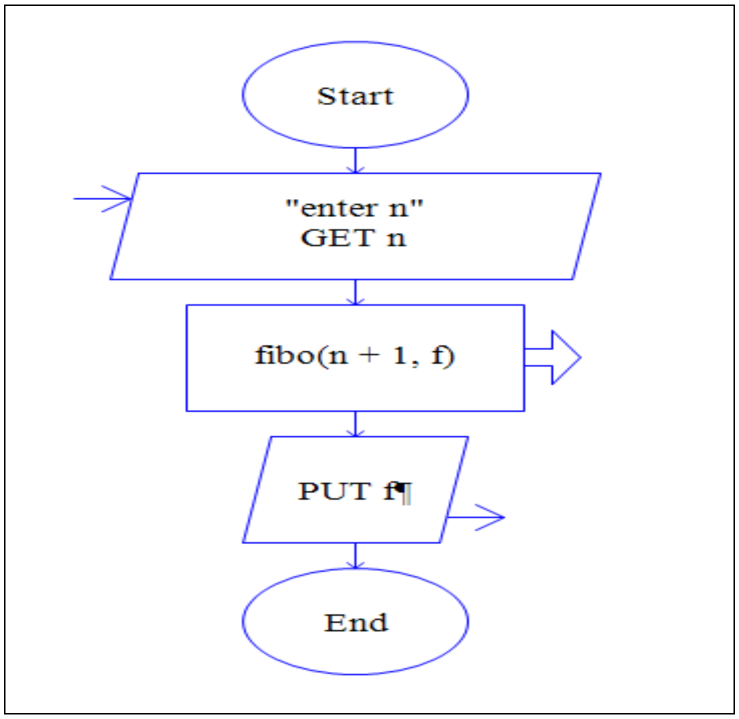
That is, newborns come from cows and infants (who come of age):
N(k) = I(k-1) + C(k-1) and cows come from infants and the surviving cows, (k) = I(k-1) + C(k-1) Now we can continue plugging in, summing previous infants plus cows to get newborns and cows for the new year, and taking previous new borns and making them infants:

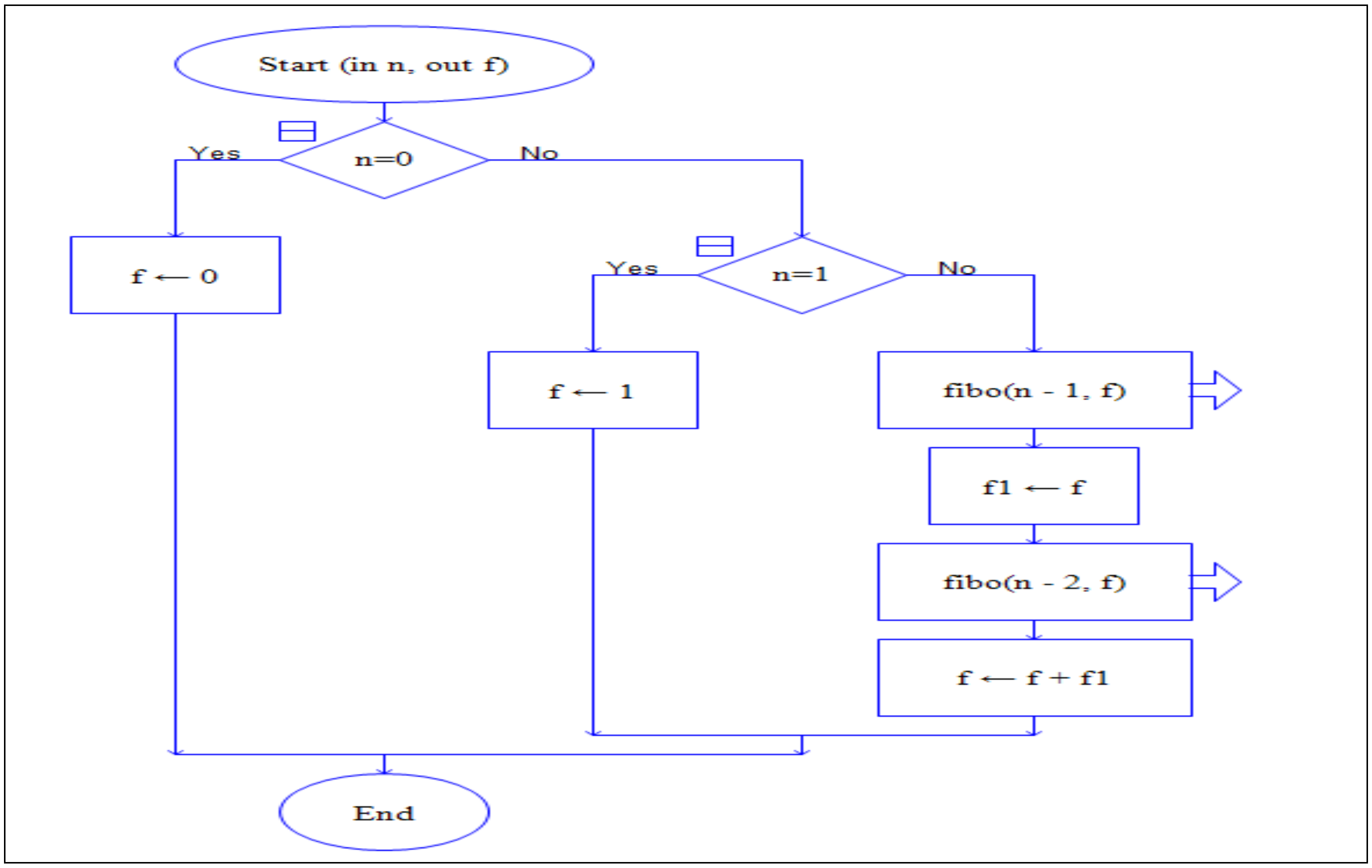
year	N	I	C
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	1
4	2	1	2
5	3	2	3
6	5	3	5
7	8	5	8

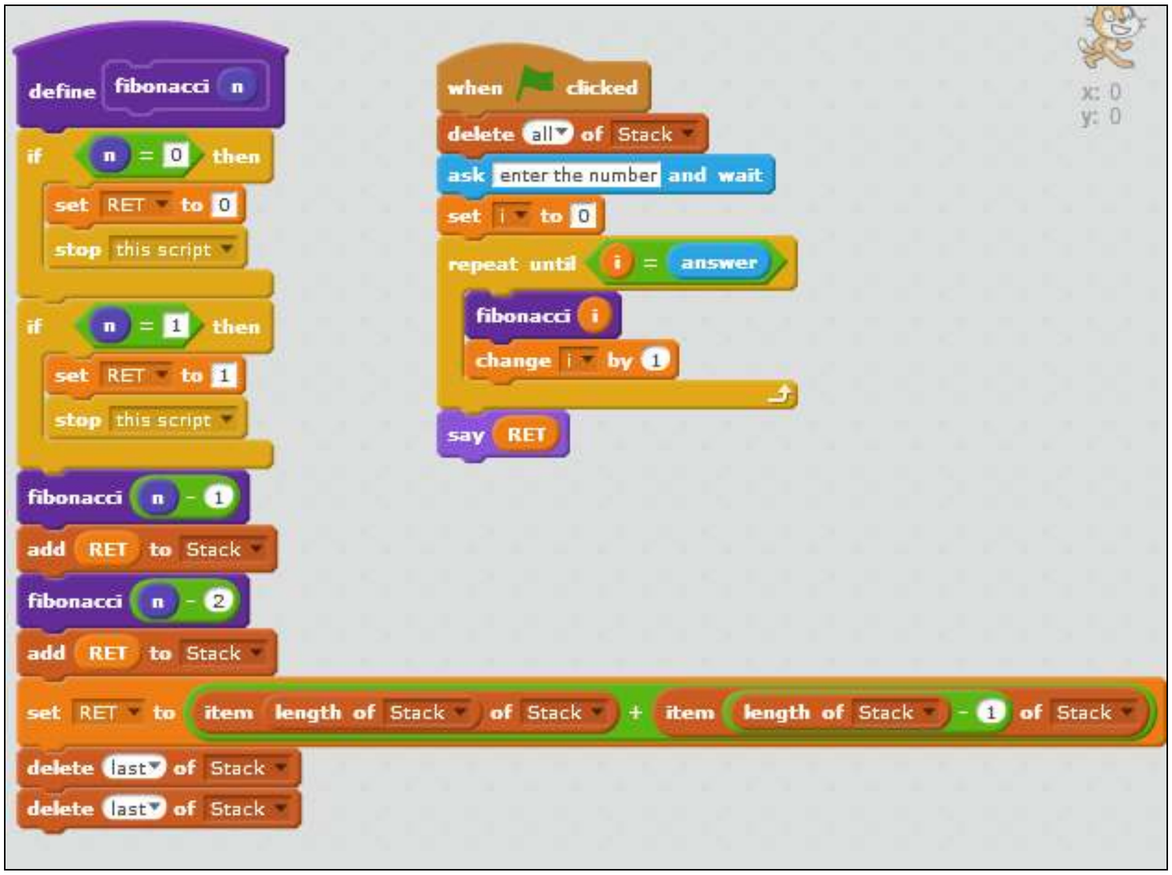
This is an application of Fibonacci numbers. After Nth year number of cows can be found out using Fibonacci number formula. The Fibonacci numbers are defined as follows,

$$F_n \equiv F_{n-2} + F_{n-1}$$

$$F_1 = F_2 = 1 \quad F_0 = 0$$


Start
"enter n"
GET n
fibo(n + 1, f)
PUT f¶
End


Start (in n, out f)
n=0
Yes
No
f ← 0
n=1
Yes
No
f ← 1
fibo(n - 1, f)
f1 ← f
fibo(n - 2, f)
f ← f + f1
End



```
define fibonacci n
if n = 0 then
    set RET to 0
    stop this script
if n = 1 then
    set RET to 1
    stop this script
fibonacci n - 1
add RET to Stack
fibonacci n - 2
add RET to Stack
set RET to item length of Stack of Stack + item length of Stack - 1 of Stack
delete last of Stack
delete last of Stack

when clicked
delete all of Stack
ask enter the number and wait
set i to 0
repeat until i = answer
    fibonacci i
    change i by 1
say RET
```

<u>Problem #2 – Permutations Calculator (nPr)</u>

The **Permutations Calculator** will find the number of sub-sets that can be taken from a larger set. However, the order of the subset now matters. The Permutations calculator will find the number of sub-sets that can be created including sub-sets of the same items in different orders.

Permutations Formula

$P(n,r) = n! / (n - r)!$

Calculate the permutations for $P(n,r) = n! / (n - r)!$. "The number of ways of obtaining an ordered subset of r elements from a set of n elements."

<u>**Permutation Problem 1:**</u> At a High School Track meet the 400 meter race has 12 contestants. The top 3 will receive points for their team. How many different permutations are there for the top 3 from the 12 contestants?

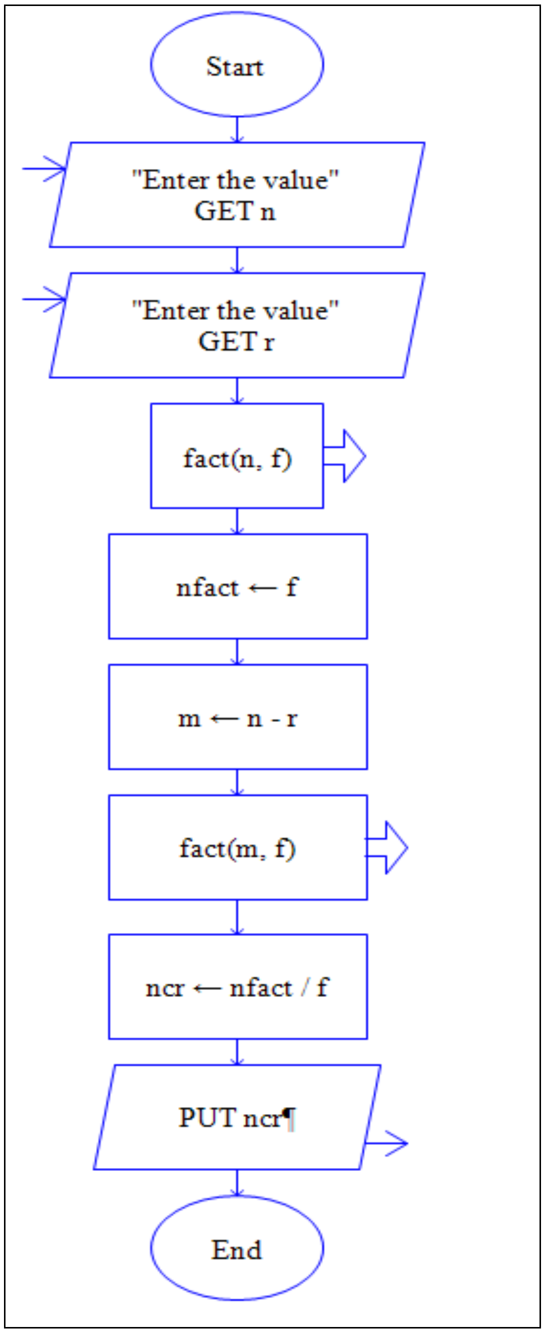
For this problem we are looking for an ordered sub-set 3 contestants (r) from the 12 contestants (n). We must calculate P(12,3) in order to find the total number of possible outcomes for the top 3.

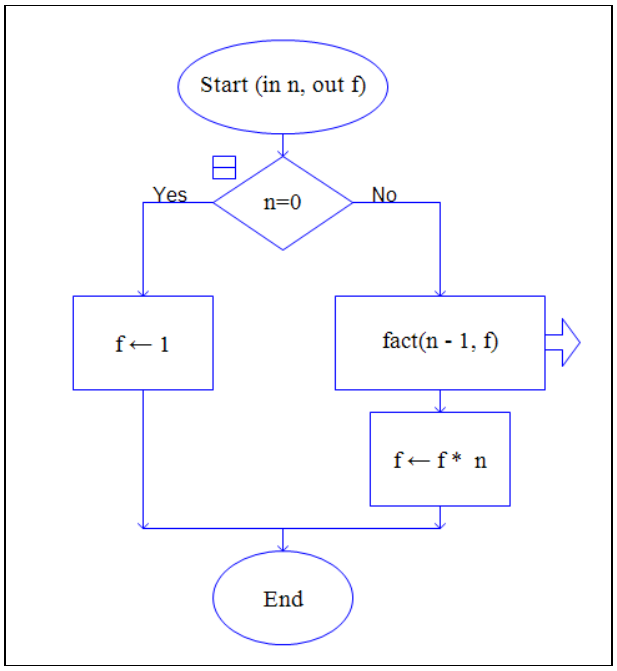
P(12,3) = 12! / (12-3)!=1,320 **Possible Outcomes**

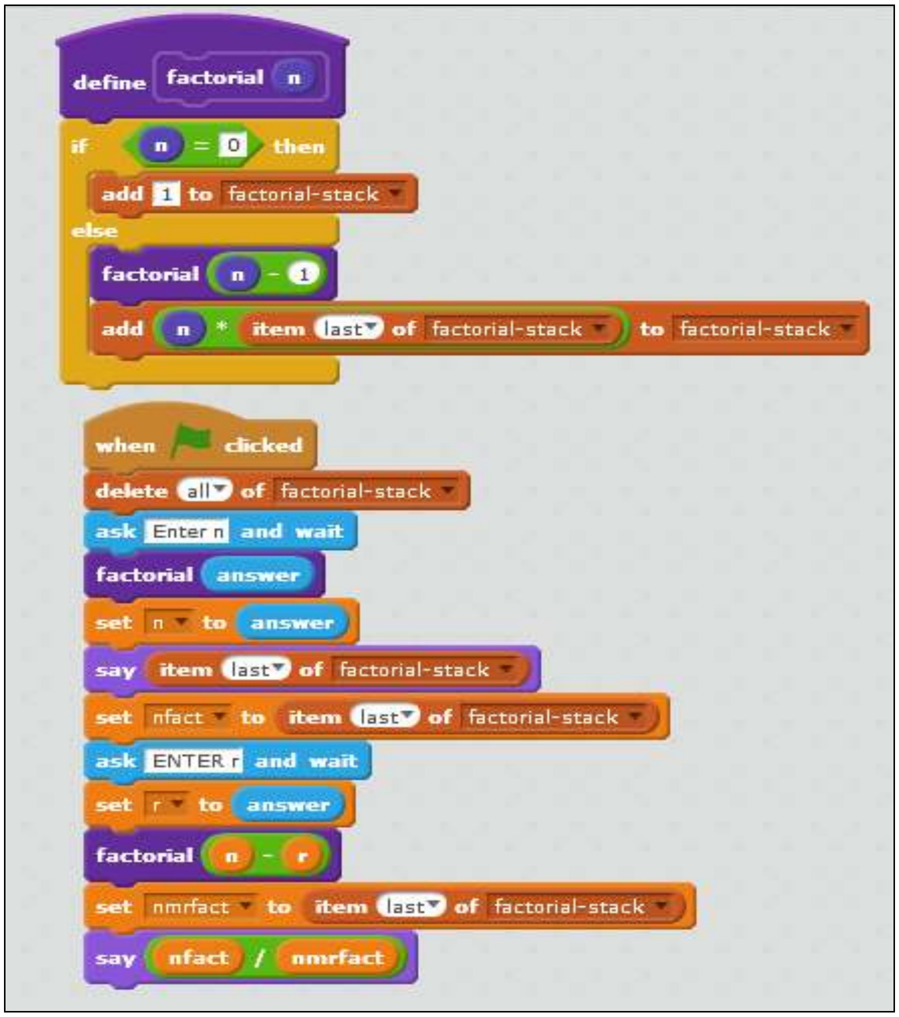
<u>Permutation Problem 2</u>: An NFL team has the 6th pick in the draft, meaning there are 5 other teams drafting before them. If the team believes that there are only 10 players that have a chance of being chosen in the top 5, how many different orders could the top 5 be chosen?

For this problem we are finding an ordered sub-set of 5 players (r) from the set of 10 players (n).

P(10,5)=10!/(10-5)!= **30,240 Possible Orders**







```
define factorial (n)
if <(n) = 0> then
    add 1 to factorial-stack
else
    factorial ((n) - 1)
    add ((n) * (item last of factorial-stack)) to factorial-stack

when flag clicked
delete all of factorial-stack
ask Enter n and wait
factorial (answer)
set n to (answer)
say (item last of factorial-stack)
set nfact to (item last of factorial-stack)
ask ENTER r and wait
set r to (answer)
factorial ((n) - (r))
set nmrfact to (item last of factorial-stack)
say ((nfact) / (nmrfact))
```

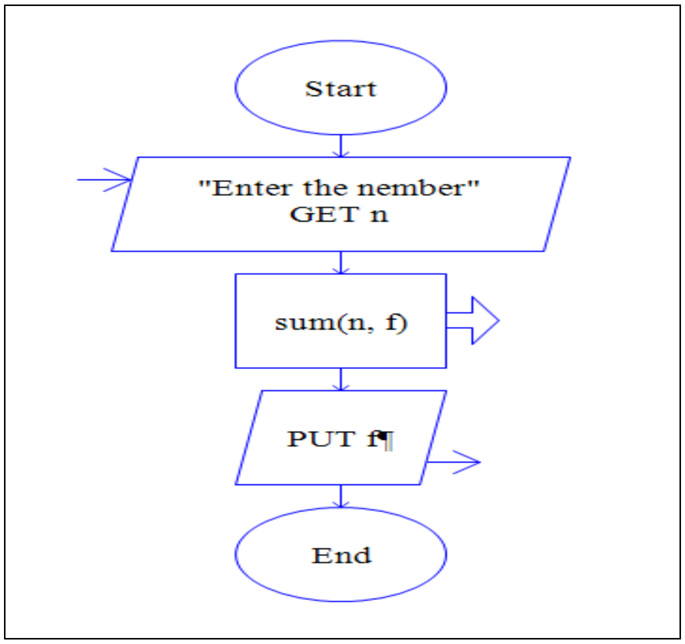
<u>Problem #3 – Sum of first n natural numbers</u>

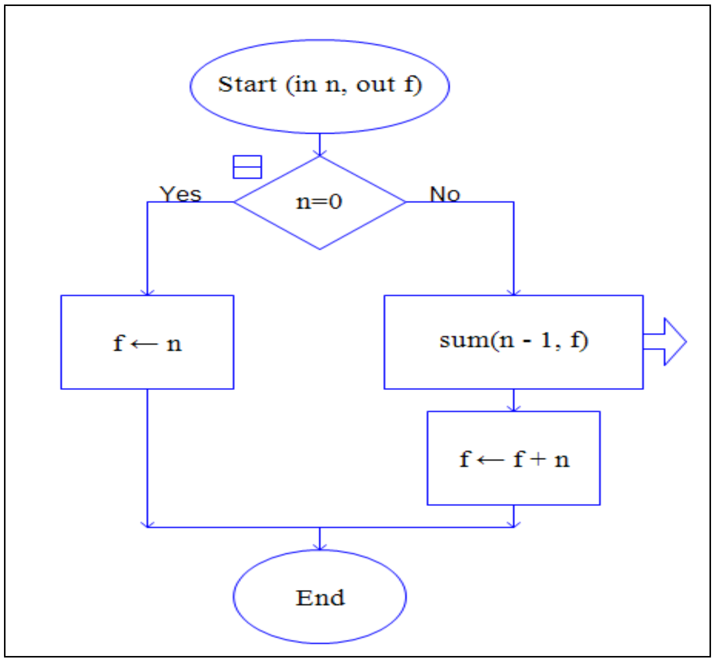
Sum of first 10 natural numbers

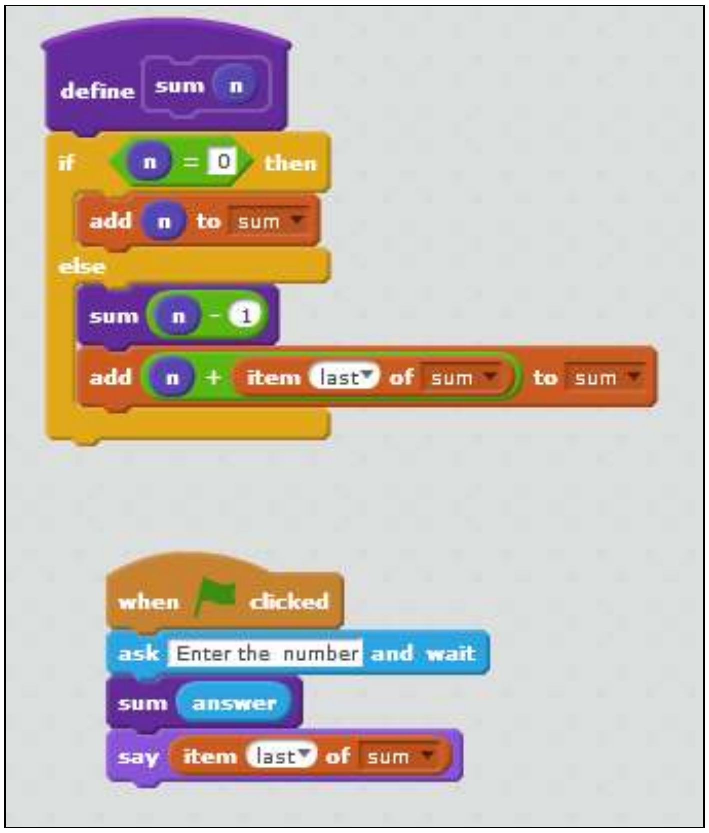
1+2+3+4+5+6+7+8+9+10

10 + sum of first 9 natural numbers $\mathbf{\overbrace{1+2+3+4+5+6+7+8+9}+10}$

19 + sum of first 9 natural numbers $\mathbf{\overbrace{1+2+3+4+5+6+7+8}+9+10}$







```
define sum (n)
if <(n) = [0]> then
    add (n) to [sum v]
else
    sum ((n) - (1))
    add ((n) + (item (last v) of [sum v])) to [sum v]

when flag clicked
ask [Enter the number] and wait
sum (answer)
say (item (last v) of [sum v])
```

Problem #4 – Student selection problem

Imagine that we have a teacher who wants to choose 10 students from a class of 25, to participate in a special project. Assuming that each student is unique in his/her own way. How many different groups of students can be chosen?

This is a classic problem using binomial coefficients that is often encountered in combinatorics for determining the amount of variety in a solution, giving insight as to the complexity of a problem. In general, if we have n items and we want to choose k items, we are looking for the solution of how many groups of k-elements can be chosen from the n elements. The expression is often written as follows and pronounced "n choose k":

$$\binom{n}{k}$$

Assuming that **n** is fixed, we can vary **k** to obtain a different answer. The simplest solutions for this problem are when **k**=0 or **k**=**n**. If **k**=0, we are asking how many groups of zero can be chosen from **n** items. The answer is **1** ... there is only one way to choose no items. Similarly, if **k**=**n** then we want to choose all items ... and there is only one way to do that as well. Also, if **k**>**n**, then there is no way to do this and the answer is zero (e.g., cannot choose 10 students from a group of 6).

Otherwise, we can express the problem as a recursive solution by examining what happens when we select a single item from the **n** items. Imagine therefore that we select one particular student that we definitely want to participate in the project (e.g., the teacher's "pet").
We need to then express the solution to the problem, taking into account that we are going to use that one student. In the remainder of the classroom, we therefore have **n**-1 students left and we still need to select **k**-1 students. Here is what is left →

$$\binom{n-1}{k-1}$$

The other situation is that we decide that we definitely DO NOT want a particular student to participate in the project (e.g., he's been a "bad boy" lately). In this case, if we eliminate that student, we still need to select **k** students, from the **n**-1 remaining. Here is what is left →

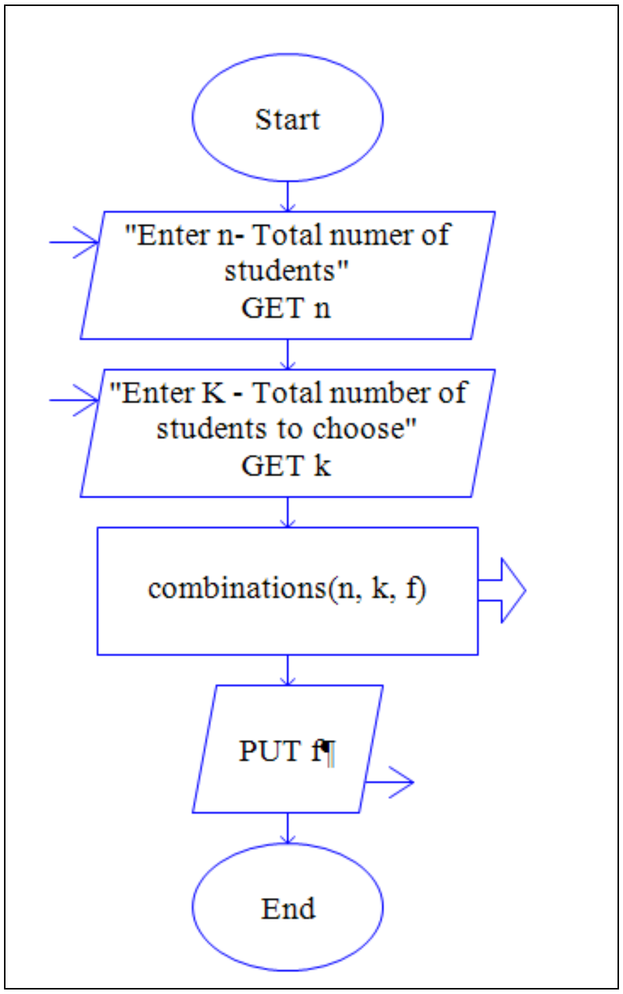
$$\binom{n-1}{k}$$

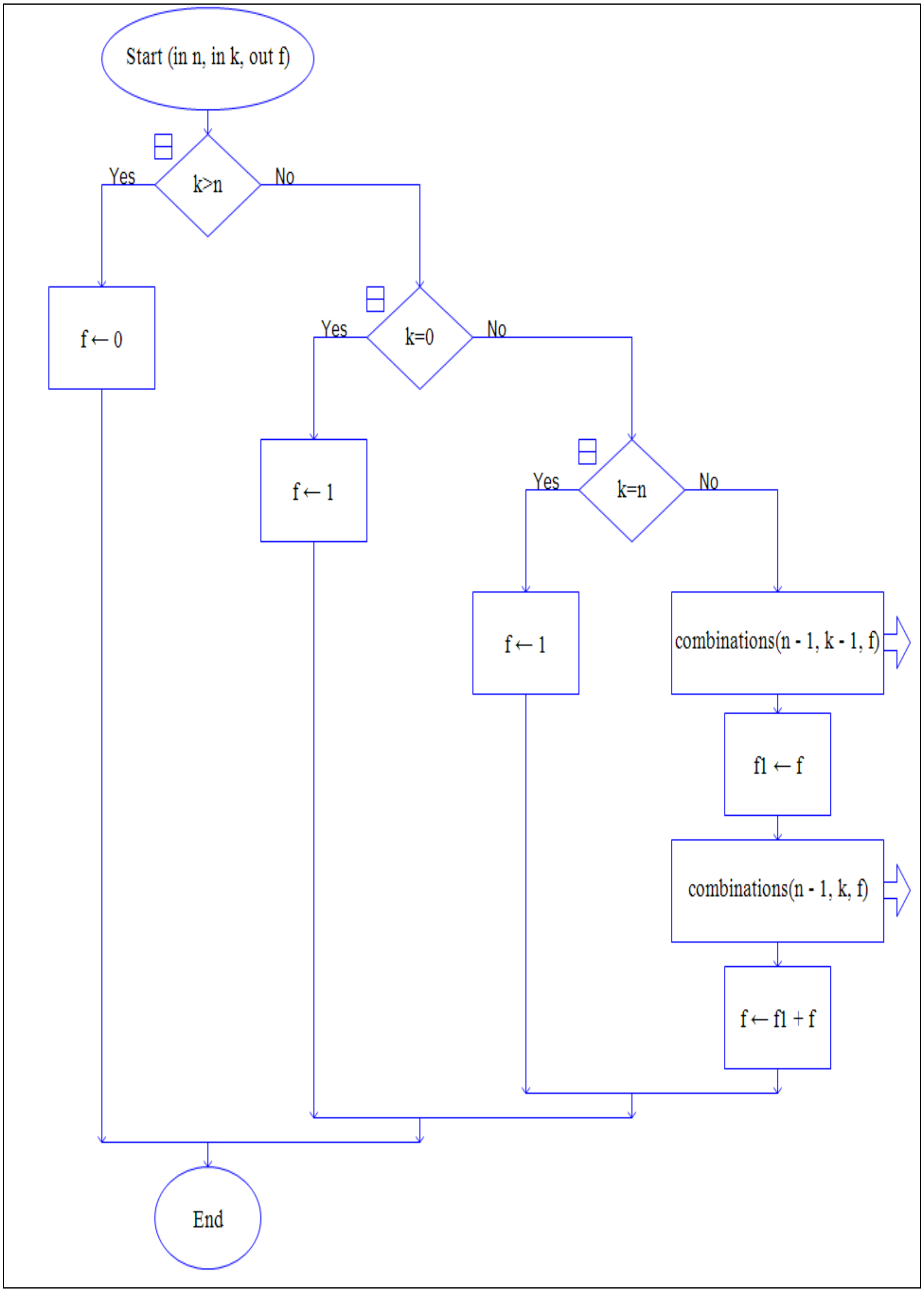
Now, I'm sure you will agree that if we examine one particular student ... then we will either select that student for the project or not. These two cases represent all possible solutions to the problem. Therefore, we can express the entire solution as:

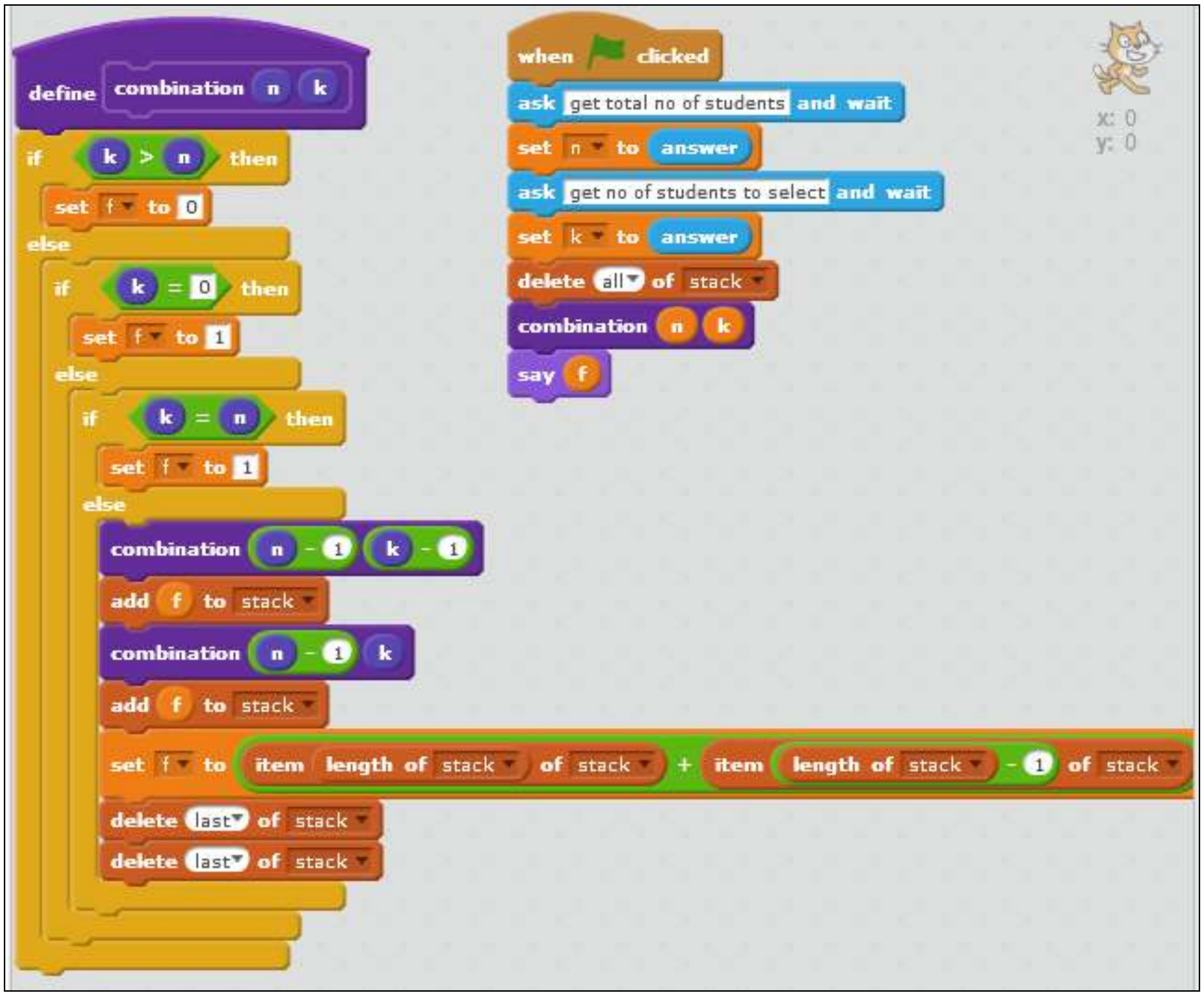
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

This is a recursive definition. The problem is of the same type and gets smaller each time (since n gets smaller and k does also in one case). This problem is interesting because there are two recursive calls to the same problem. That is, the problem branches off twice recursively.

```
Function: StudentCombinations
        n: total number of students
        k: number of students to choose
    1. if (k > n) then
            a. return 0
    2. if (k is 0) then
            a. return 1
    3. if (k is n) then
            a. return 1
    4. return StudentCombinations(n-1,k-1) + StudentCombinations(n-1,k)
```


Start
"Enter n- Total numer of students"
GET n
"Enter K - Total number of students to choose"
GET k
combinations(n, k, f)
PUT f¶
End


Start (in n, in k, out f)
k>n
Yes
No
f ← 0
k=0
Yes
No
f ← 1
k=n
Yes
No
f ← 1
combinations(n - 1, k - 1, f)
f1 ← f
combinations(n - 1, k, f)
f ← f1 + f
End



```
define combination (n) (k)
if <(k) > (n)> then
    set f to 0
else
    if <(k) = 0> then
        set f to 1
    else
        if <(k) = (n)> then
            set f to 1
        else
            combination ((n) - 1) ((k) - 1)
            add (f) to stack
            combination ((n) - 1) (k)
            add (f) to stack
            set f to ((item (length of stack) of stack) + (item ((length of stack) - 1) of stack))
            delete (last) of stack
            delete (last) of stack

when flag clicked
ask [get total no of students] and wait
set n to (answer)
ask [get no of students to select] and wait
set k to (answer)
delete (all) of stack
combination (n) (k)
say (f)
```

5.2. Factoring

In mathematics, **factorization** or **factoring** is the decomposition of an object (for example, a number, a polynomial, or a matrix) into a product of other objects, or *factors*, which when multiplied together give the original.

For example, the number 15 factors into primes as 3 × 5,

and the polynomial $x^2 - 4$ factors as $(x - 2)(x + 2)$.

In all cases, a product of simpler objects is obtained. The aim of factoring is usually to reduce something to "basic building blocks", such as numbers to prime numbers, or polynomials to irreducible polynomials.

Sample Problems

Problem #1 – Smallest divisor of a number

To find smallest divisor of an integer n is very straight forward. Just take the numbers 2, 3, 4… n and divide n with each of number in ascending order. But not efficient method, because it need more iteration (loops).

To efficiently calculate this problem we must reduce the loops by various techniques and are follows,

· All exact divisors of an integer are paired. For example an integer 24 have following divisors 2, 3, 4, 6, 8, 12. And each divisor has pair.

Here 2 and 12 are paired. In other word they are linked because

24/2 = 12

2, 12 are smaller and bigger factors respectively.

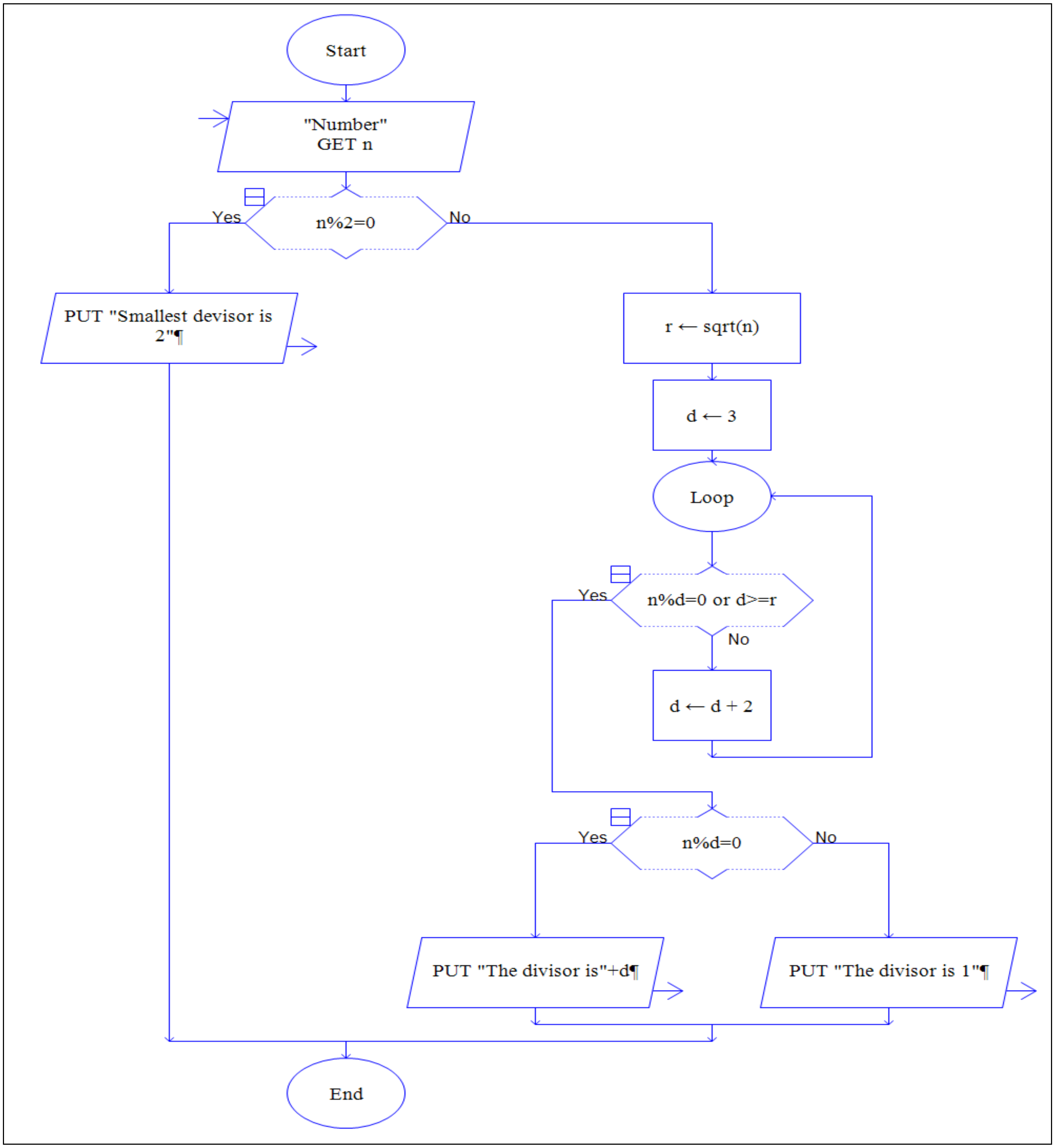
Same way 3 and 8 are paired, i.e. (24/3 = 8 ;) and so on.

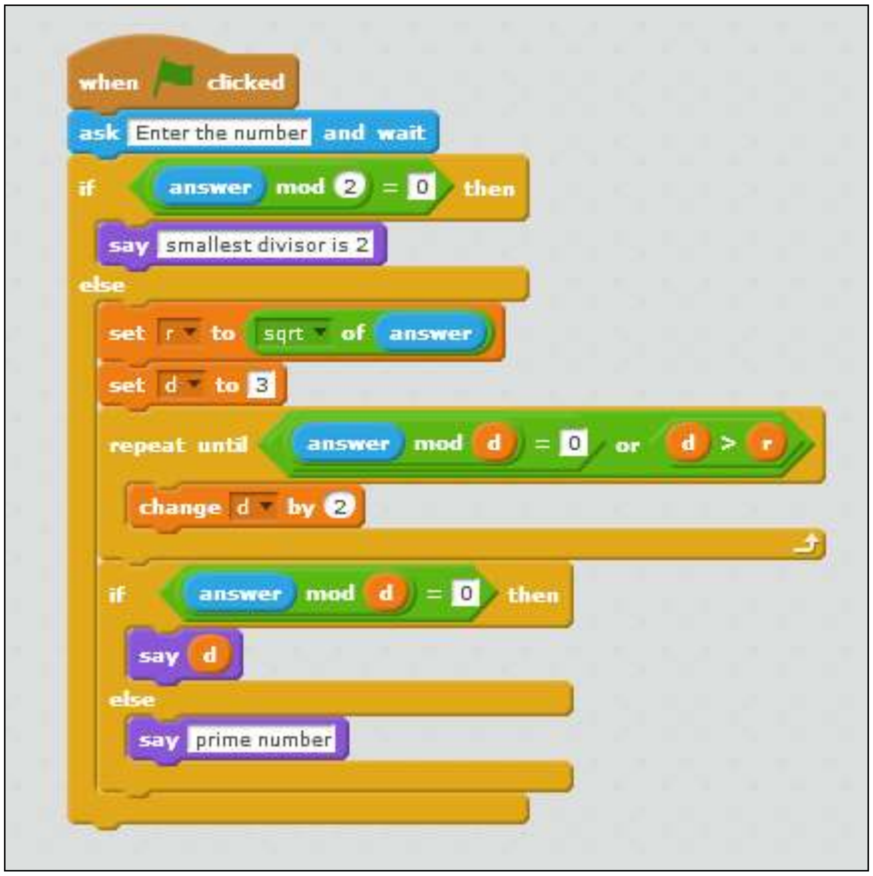
Smaller factor	Bigger factor
2	12
3	8
4	6

Here we can see that smallest divisor 2 is paired with largest divisor 12. 2nd smallest divisor is paired with 2nd largest divisor. And so on. If the integer is 36 we can see divisor 6 is paired with 6 itself. In general case we have to consider only less than or equal to biggest smaller divisor, now the problem!. How can we take that limit? Simple take √n, all smaller divisors of an integer are less √n. If n is an even number then its smallest divisor is 2. Now we need consider only odd numbers. If we get n is the smallest divisor of n then it's a prime, i.e. one is smallest divisor.

Algorithm:

```
1. Establish the integer n
2. If n is even then 2 is smallest divisor.
3. Else
   a. Compute r = √n;
   b. Initialize divisor d = 3;
   c. While not an exact divisor and r ≠ √n
        i. Generate next divisor d in odd sequence by d = d +2;
   d. If d is exact divisor and d ≠ n, then d is the smallest divisor of n
        i. Else n is prime
```


Start
"Number"
GET n
n%2=0
Yes
No
PUT "Smallest devisor is 2"¶
r ← sqrt(n)
d ← 3
Loop
n%d=0 or d>=r
Yes
No
d ← d + 2
n%d=0
Yes
No
PUT "The divisor is"+d¶
PUT "The divisor is 1"¶
End



```
when [flag] clicked
ask [Enter the number] and wait
if <(answer) mod (2) = [0]> then
    say [smallest divisor is 2]
else
    set [r] to (sqrt of (answer))
    set [d] to [3]
    repeat until <<(answer) mod (d) = [0]> or <(d) > (r)>>
        change [d] by (2)
    end
    if <(answer) mod (d) = [0]> then
        say (d)
    else
        say [prime number]
    end
end
```

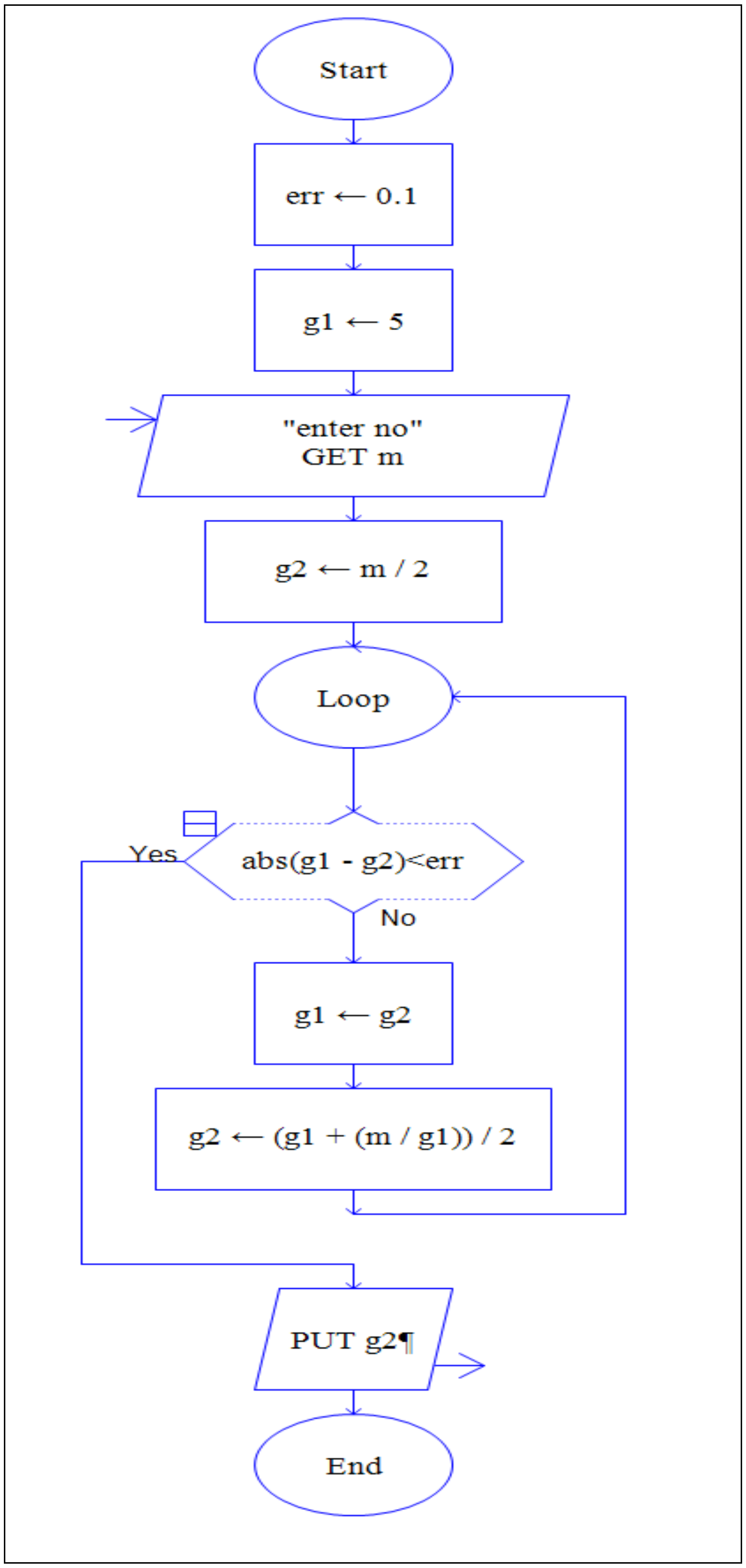
<u>Problem #2 – Square root of a number</u>

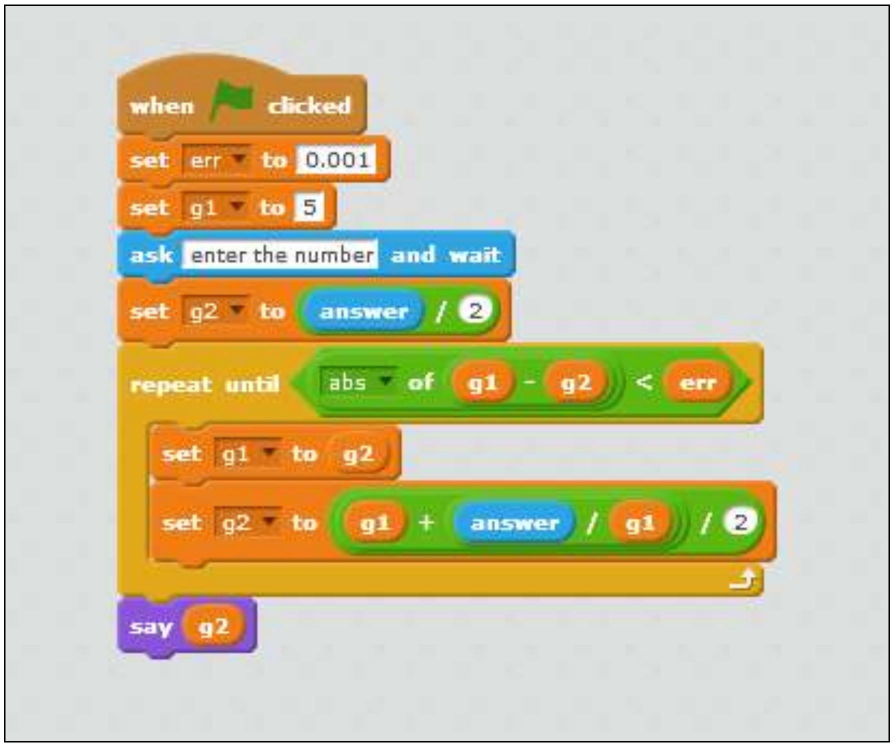
The square root of a positive number **b** can be computed with Newton's formula:

$$\text{New } x = \frac{1}{2}\left(x + \frac{b}{x}\right)$$

where **x** above starts with a "reasonable" guess. In fact, you can always start with **b** or some other value, say **1**.

With **b** and a guess value **x**, a new guess value is computed with the above formula. This process continues until the new guess value and the current guess value are very close. In this case, either one can be considered as an approximation of the square root of **b**.


Start
err ← 0.1
g1 ← 5
"enter no"
GET m
g2 ← m / 2
Loop
abs(g1 - g2)<err
Yes
No
g1 ← g2
g2 ← (g1 + (m / g1)) / 2
PUT g2¶
End



```
when [flag] clicked
set err to 0.001
set g1 to 5
ask enter the number and wait
set g2 to (answer / 2)
repeat until < abs of (g1 - g2) < err >
    set g1 to g2
    set g2 to ((g1 + (answer / g1)) / 2)
say g2
```

<u>Problem #3 – GCD of two numbers</u>

The greatest common divisor (GCD) of two integers *a* and *b* is defined to be the largest integer that divides both *a* and *b* with no remainder. For example, the GCD of 16 and 28 is 4. One way to find the GCD of two integers is to factor them and search for common factors, but there is a famous algorithm that is much more efficient.The idea of the algorithm is based on the observation that, if *r* is the remainder when *a* is divided by *b*, then the common divisors of *a* and *b* are precisely the same as the common divisors of *b* and *r*. Thus, we can use the equation

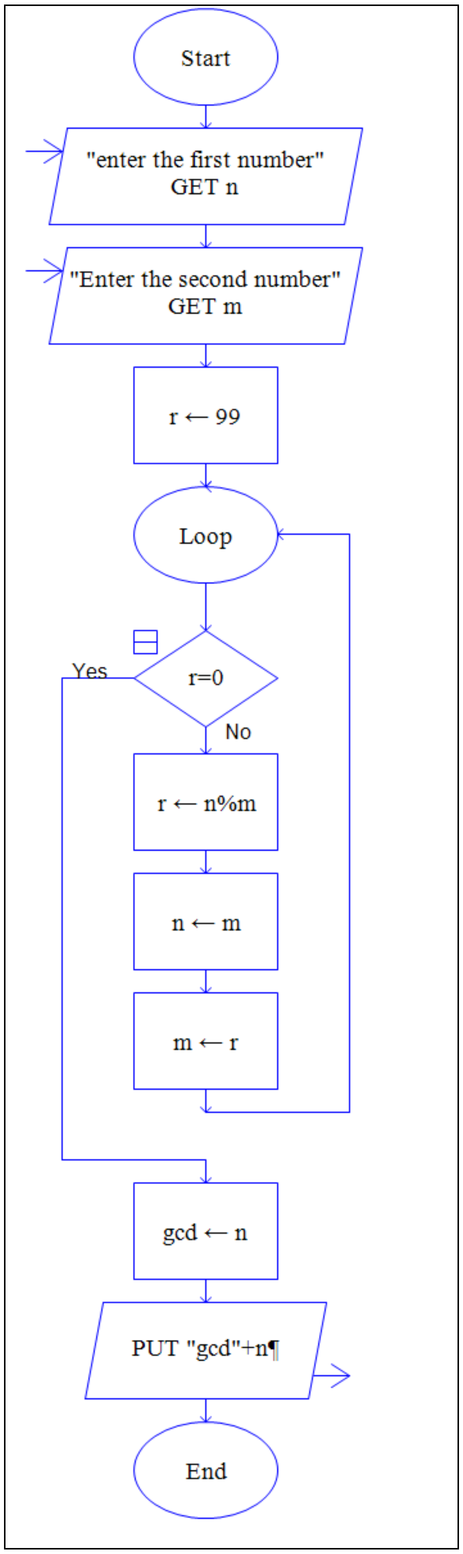
$$\mathrm{GCD}(a,b) = \mathrm{GCD}(b,r)$$

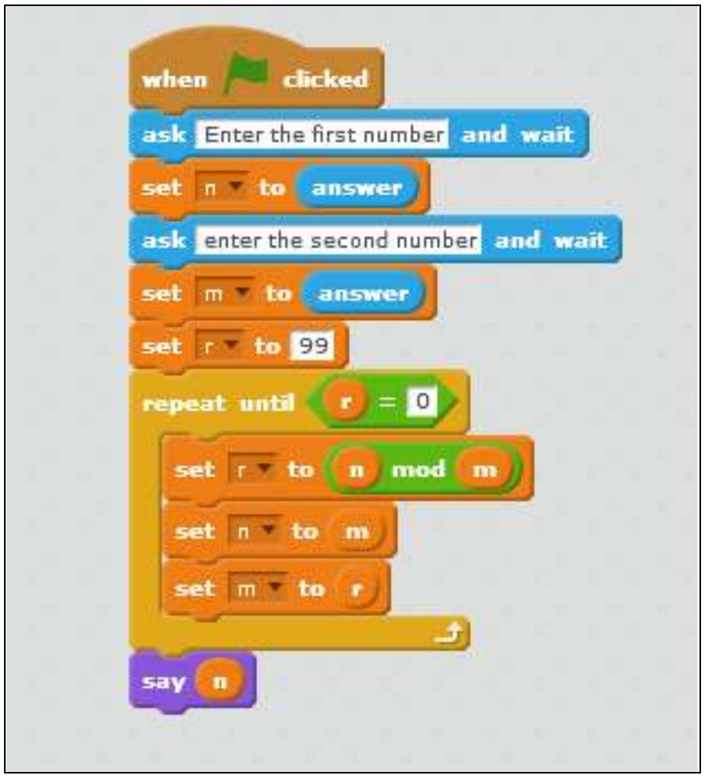
to successively reduce the problem of computing a GCD to the problem of computing the GCD of smaller and smaller pairs of integers. For example,

$$\begin{aligned} \mathrm{GCD}(206,40) &= \mathrm{GCD}(40,6) \\ &- \mathrm{GCD}(6,4) \\ &= \mathrm{GCD}(4,2) \\ &= \mathrm{GCD}(2,0) \\ &= 2 \end{aligned}$$

reduces GCD(206,40) to GCD(2,0), which is 2. It is possible to show that starting with any two positive integers and performing repeated reductions will always eventually produce a pair where the second number is 0. Then the GCD is the other number in the pair

`


Start
"enter the first number"
GET n
"Enter the second number"
GET m
r ← 99
Loop
r=0
Yes
No
r ← n%m
n ← m
m ← r
gcd ← n
PUT "gcd"+n¶
End



```
when [flag] clicked
ask [Enter the first number] and wait
set n to (answer)
ask [enter the second number] and wait
set m to (answer)
set r to 99
repeat until <(r) = 0>
    set r to ((n) mod (m))
    set n to (m)
    set m to (r)
say (n)
```

5.3. Exercises

LAQ5.1. Computing a positive integer power of a number. Consider a^n : If n = 0, a^n is 1 (by definition) If n > 0, a^n is a * a^{n-1}

LAQ5.2. Suppose Selena would like to buy a new smart phone that costs $X. She borrows the money from her parents and with the money she earns from a part-time job, she pays them $Y per week. How long will it take her to pay off the loan? Write a recursive function for the above calculation.

LBQ5.3. In a machine 3 sensor are fitted. The readings are single digit numbers and in display these numbers are displayed together as a single number. The actual result we want is the sum of all the readings. Write a recursive method to find the sum of all sensor values.

LBQ5.4. The equation used to describe the output of a discrete-time filter is in terms of a weighted combination of the input and previous output samples. For example a first-order filter may have the following difference equation y(m) = a y(m −1) + x(m) , where x(m) is the filter input, y(m) is the filter output and a is the filter coefficient. Find the value of y(m) using recursion. Assume the initial condition y(0)=1, a=10 and x(m)=1 for all m.

LAQ5.5. Denis's school is planning a hot dog lunch to raise money for sports equipment. Based on the last hot dog lunch, the students expect to sell at least 100 hot dogs. Wieners come in packages of 12 and buns come in packages of 8. How many packages of wieners and buns should the students buy to ensure that the numbers of wieners and buns are equal?

LBQ5.6. Your school plans to make and sell homemade ice cream. They will be having a contest for the container design. One of the design requirements states that the length, width, and height must be whole numbers greater than 1, and the container must be designed to hold about 1 gallon of ice cream. One gallon of ice cream is approximately 210 cubic inches. Determine the possible dimensions of the container.

LAQ5.7. Two bikers are riding a circular path. The first rider completes a round in 12 minutes. The second rider completes a round in 18 minutes. If they both started at the same place and time and go in the same direction, after how many minutes will they meet again at the starting point?

LBQ5.8. Kiara baked 30 oatmeal cookies and 48 chocolate chip cookies to package in plastic containers for her teacher friends at school. She wants to divide the cookies into identical containers so that each container has the same number of each kind of cookie. If she wants each container to have the greatest number of cookies possible, how many plastic containers does she need?

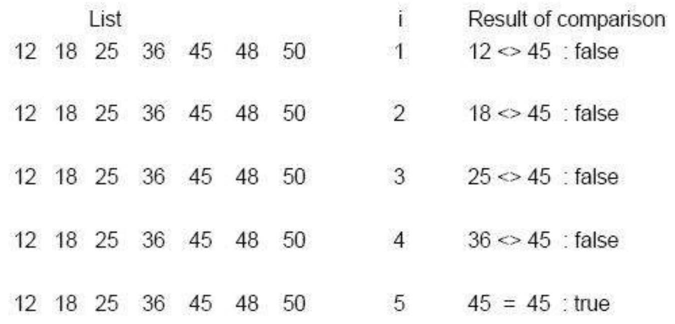
6. SEARCHING AND SORTING TECHNIQUES

6.1. Searching

Linear Search (Sequential Search)

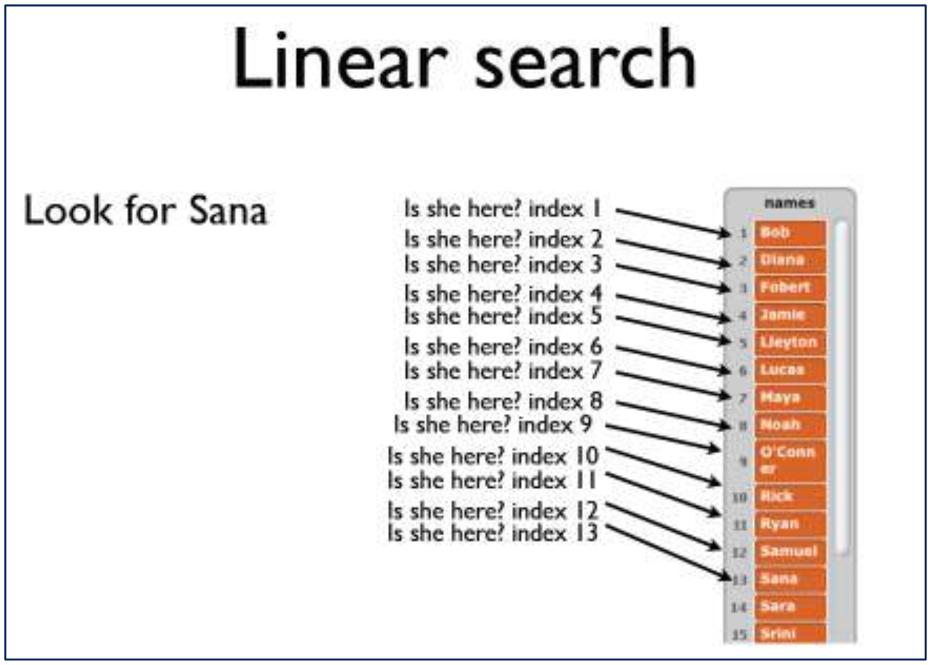
Every element in the data set is examined in the order presented until the value being searched for is found then the technique is called as known as linear search. The linear search is also known as the sequential search.

Assume the element 45 is to be searched from a sequence of sorted elements 12, 18, 25, 36, 45, 48, 50. The Linear search starts from the first element 12, since the value to be searched is not 12 (value 45), the next element 18 is compared and is also not 45, by this way all the elements before 45 are compared and when the index is 5, the element 45 is compared with the search value and is equal, hence the element is found and the element position is 5. The sequence of steps is as follows.



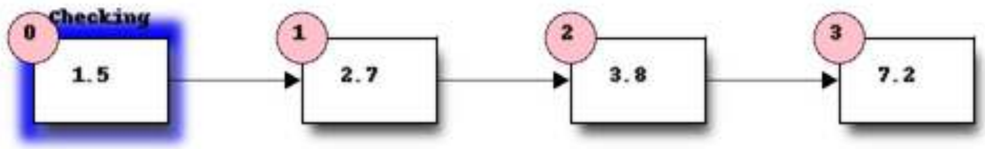
List	i	Result of comparison
12 18 25 36 45 48 50	1	12 <> 45 : false
12 18 25 36 45 48 50	2	18 <> 45 : false
12 18 25 36 45 48 50	3	25 <> 45 : false
12 18 25 36 45 48 50	4	36 <> 45 : false
12 18 25 36 45 48 50	5	45 = 45 : true

Example-1

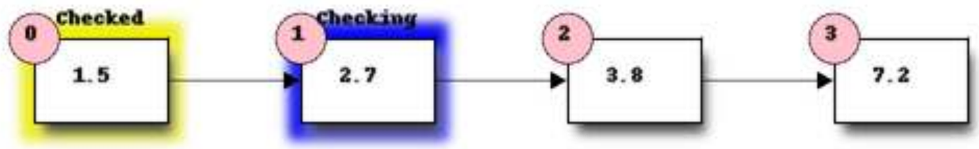


Example-2

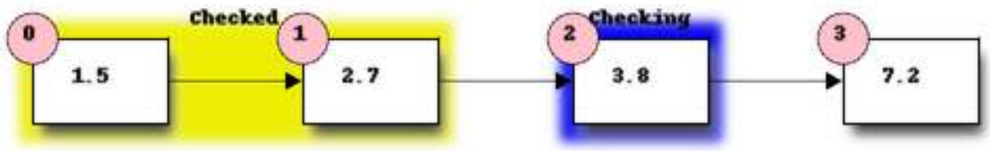
For example, suppose that we want to find the number 3.8 in the following list:



We start with the first element, and perform a comparison to see if its value is the value that we want. In this case, 1.5 is not equal to 3.8, so we move onto the next element:



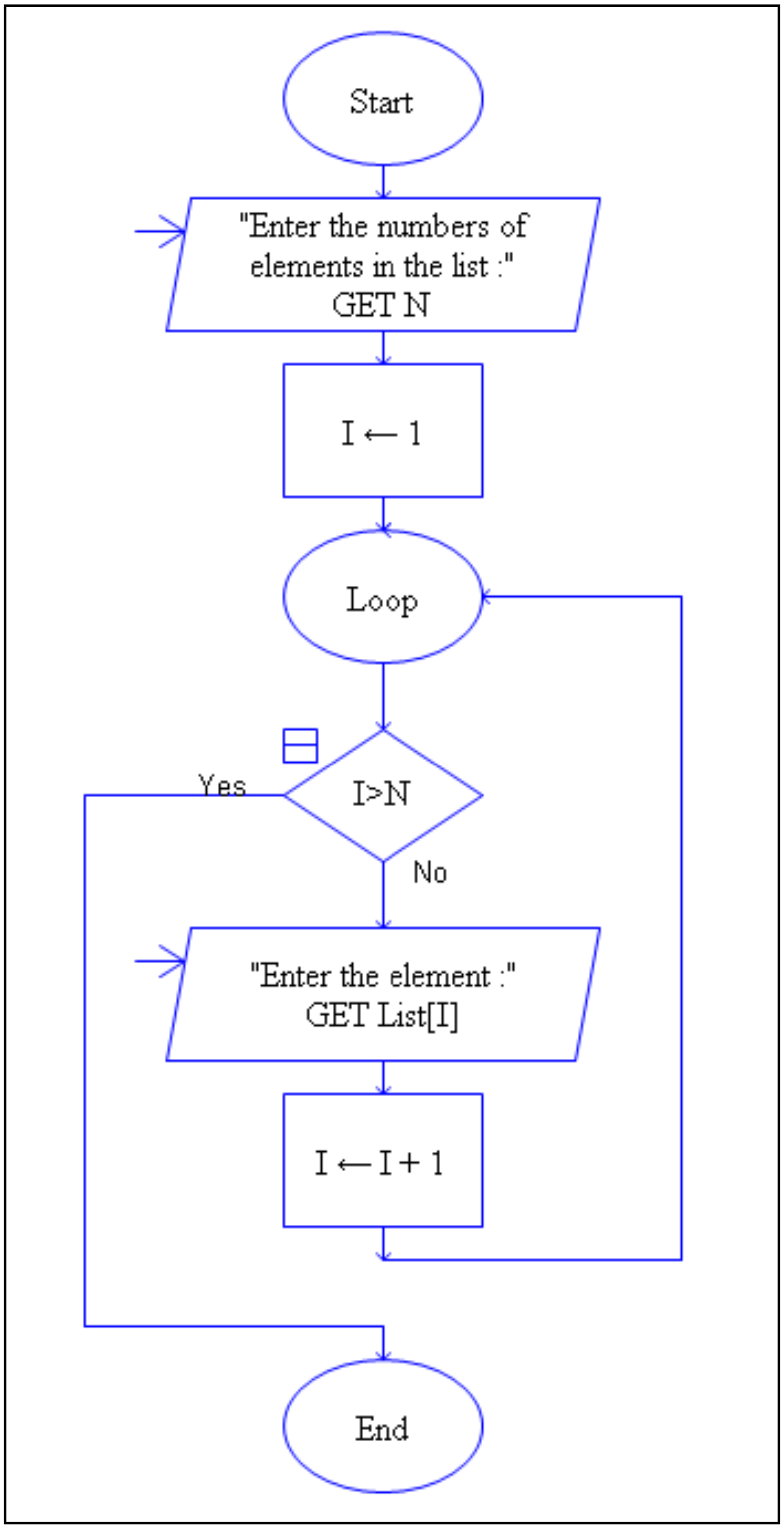
We perform another comparison, and see that 2.7 is also not equal to 3.8, so we move onto the next element:



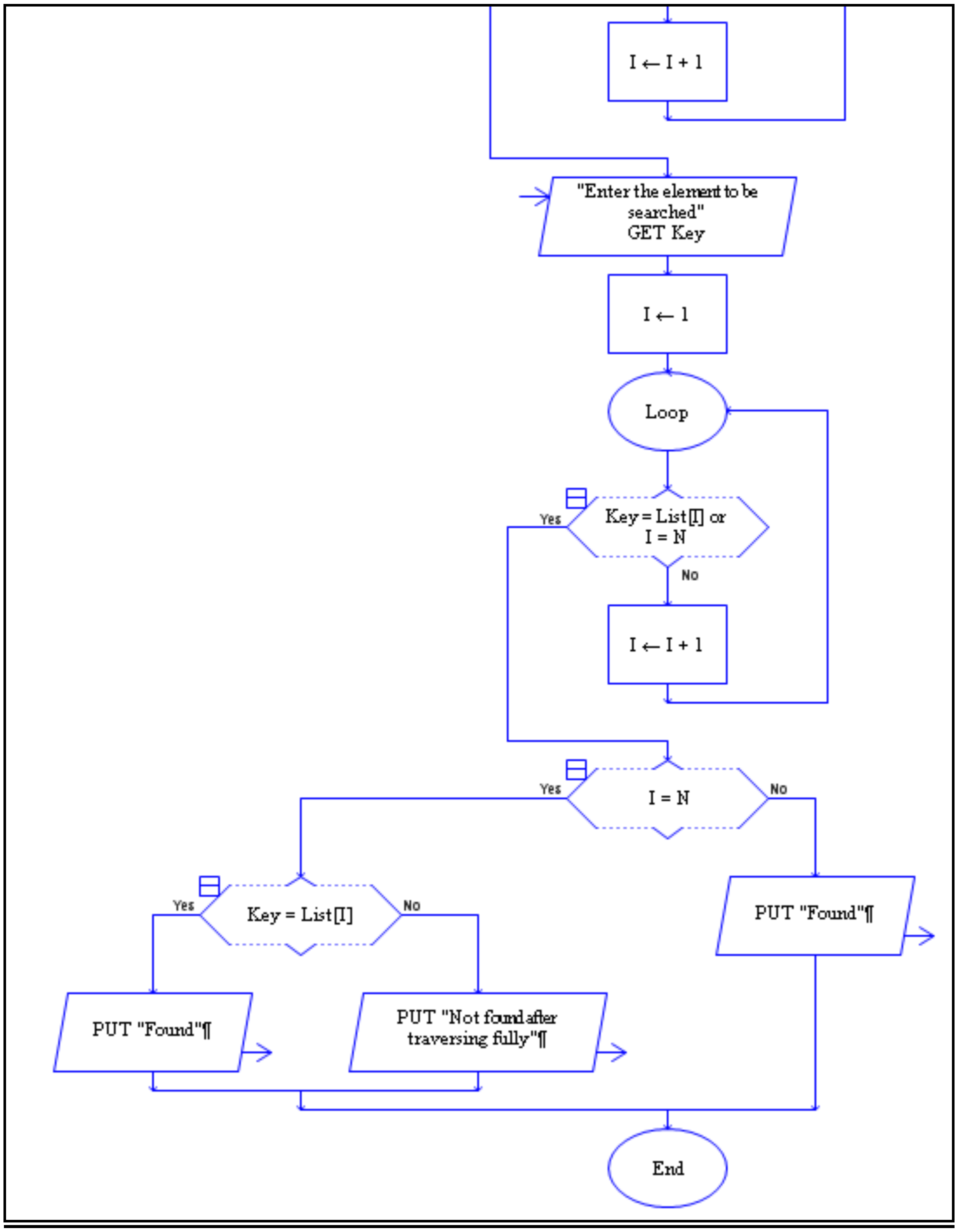
We perform another comparison and determine that we have found the correct element. Now we can end the search and return the position of the element (index 2).

<u>How to perform Linear Search in RAPTOR?</u>

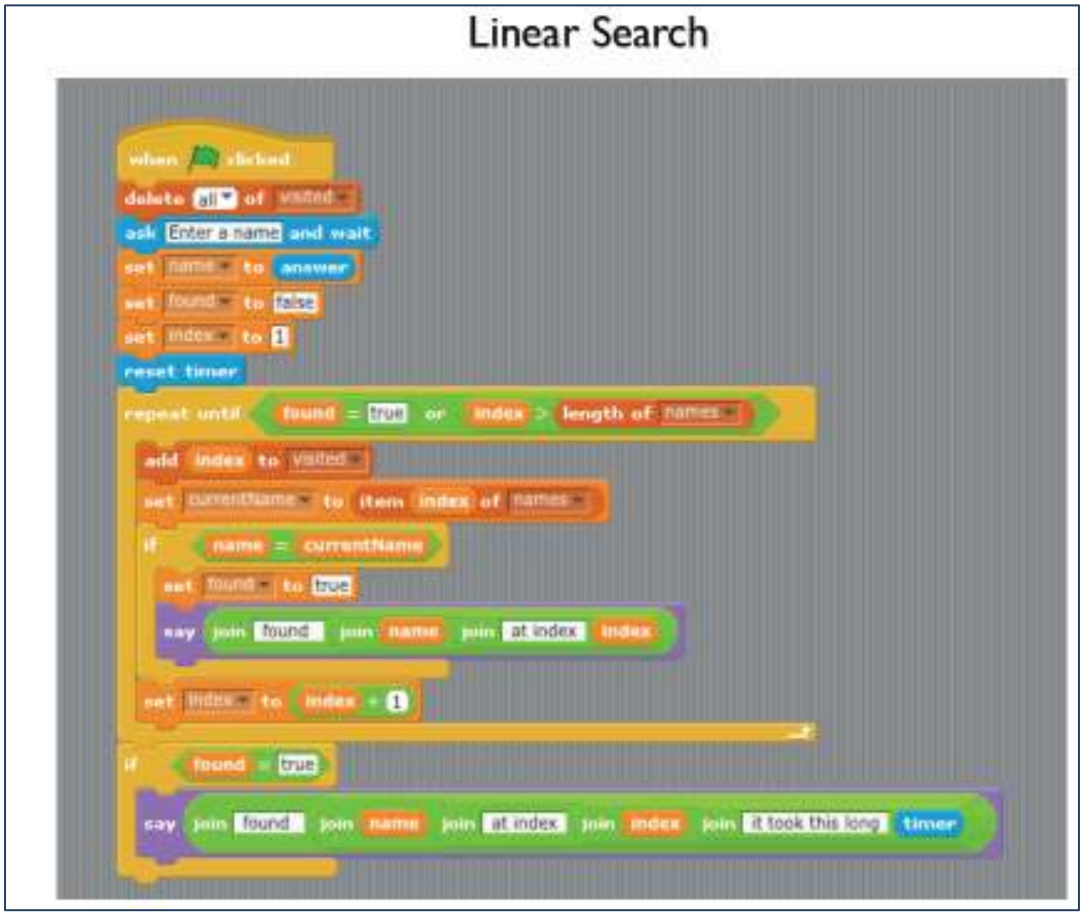
<u>Step 1:</u> *Let's try a flow chart for receiving the set of elements from the user in RAPTOR…*


Start
"Enter the numbers of elements in the list :" GET N
I ← 1
Loop
I>N
Yes
No
"Enter the element :" GET List[I]
I ← I + 1
End

Step 2: *Now let's look at the search in sequential order for a key element from the list containing set of elements in RAPTOR…*



<u>How to perform Linear Search in Scratch?</u>



Linear Search

```
when [flag] clicked
delete all of visited
ask Enter a name and wait
set name to answer
set found to false
set index to 1
reset timer
repeat until < <found = true> or <index > length of names> >
    add index to visited
    set currentName to item index of names
    if <name = currentName>
        set found to true
        say join found join name join at index index
    set index to index + 1
if <found = true>
    say join found join name join at index join index join it took this long timer
```

<u>Binary Search</u>

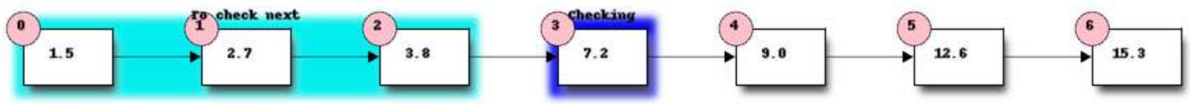
Binary search is a more efficient search algorithm which relies on the elements in the list being sorted. We apply the same search process to progressively smaller sub-lists of the original list, starting with the whole list and approximately halving the search area every time.

We first check the *middle* element in the list.

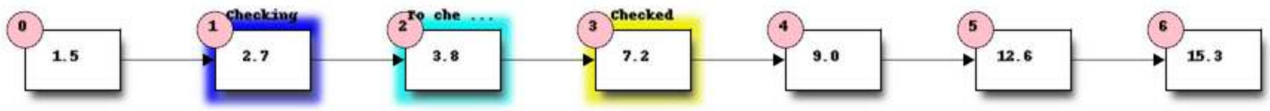
- If it is the value we want, we can stop.
- If it is *higher* than the value we want, we repeat the search process with the portion of the list *before* the middle element.
- If it is *lower* than the value we want, we repeat the search process with the portion of the list *after* the middle element.

<u>Example-1</u>

For example, suppose that we want to find the value 3.8 in the following list of 7 elements:

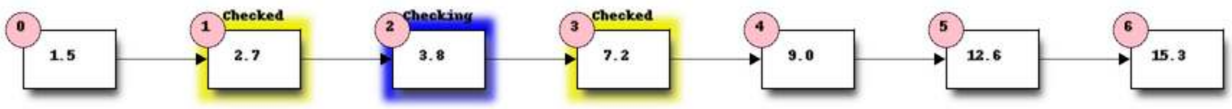


First, we compare the element in the middle of the list to our value. 7.2 is *bigger* than 3.8, so we need to check the first half of the list next.



Now the first half of the list is our new list to search. We compare the element in the middle of this list to our value. 2.7 is *smaller* than 3.8, so we need to search the *second half* of this sublist next.

The second half of the last sub-list is just a single element, which is also the middle element. We compare this



element to our value, and it is the element that we want.

<u>Example-2</u>

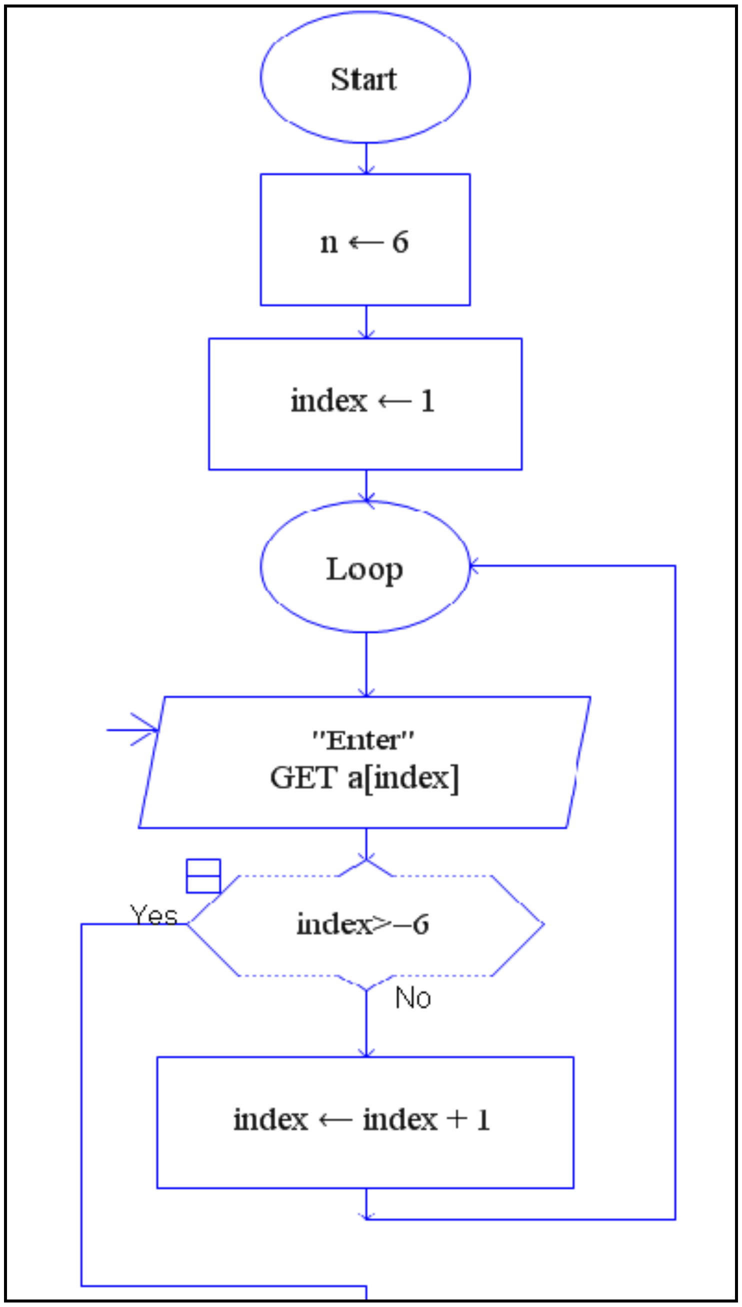
The list to be searched: L = 1 3 4 6 8 9 11. The value to be found: X = 4.

Compare X to 6. X is smaller. Repeat with L = 1 3 4.

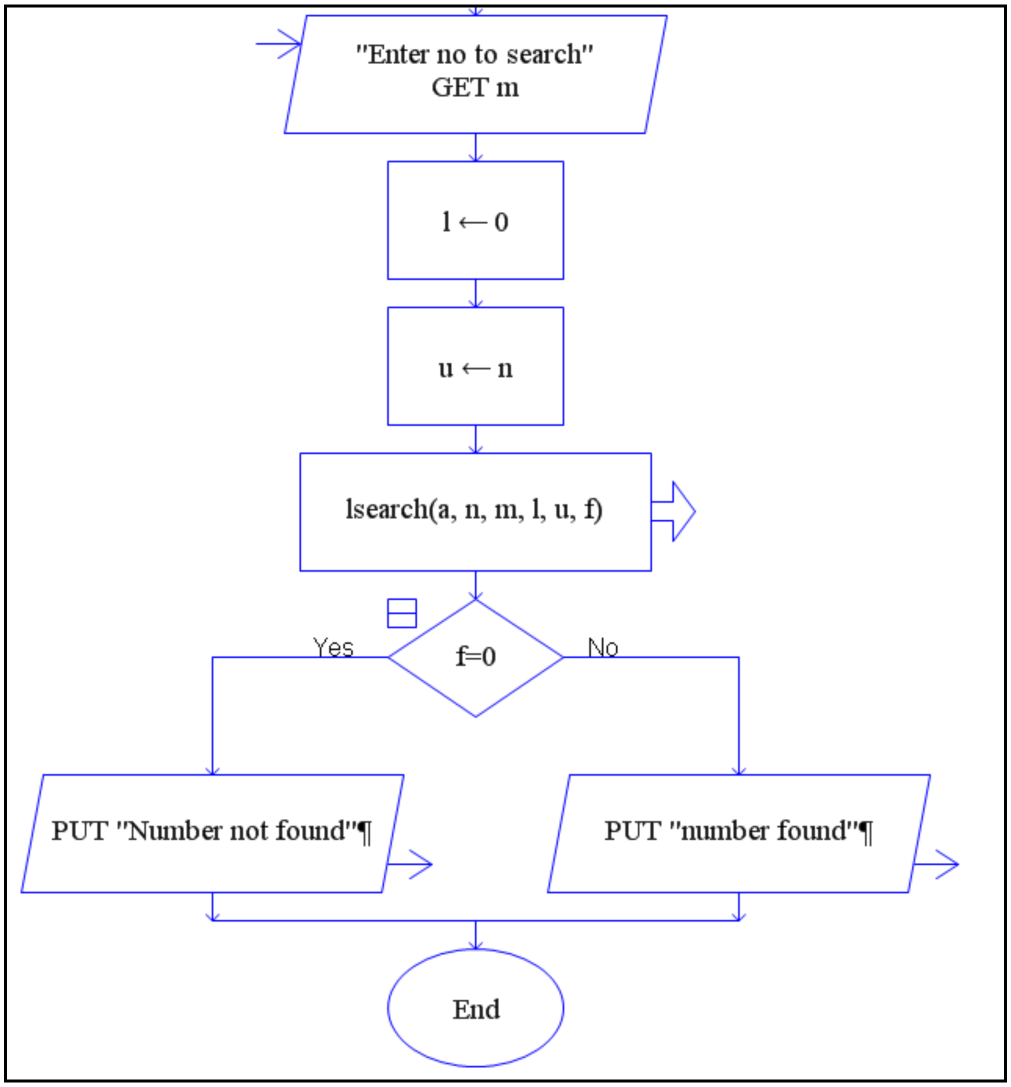
Compare X to 3. X is bigger. Repeat with L = 4.

Compare X to 4. They are equal. We're done, we found X.

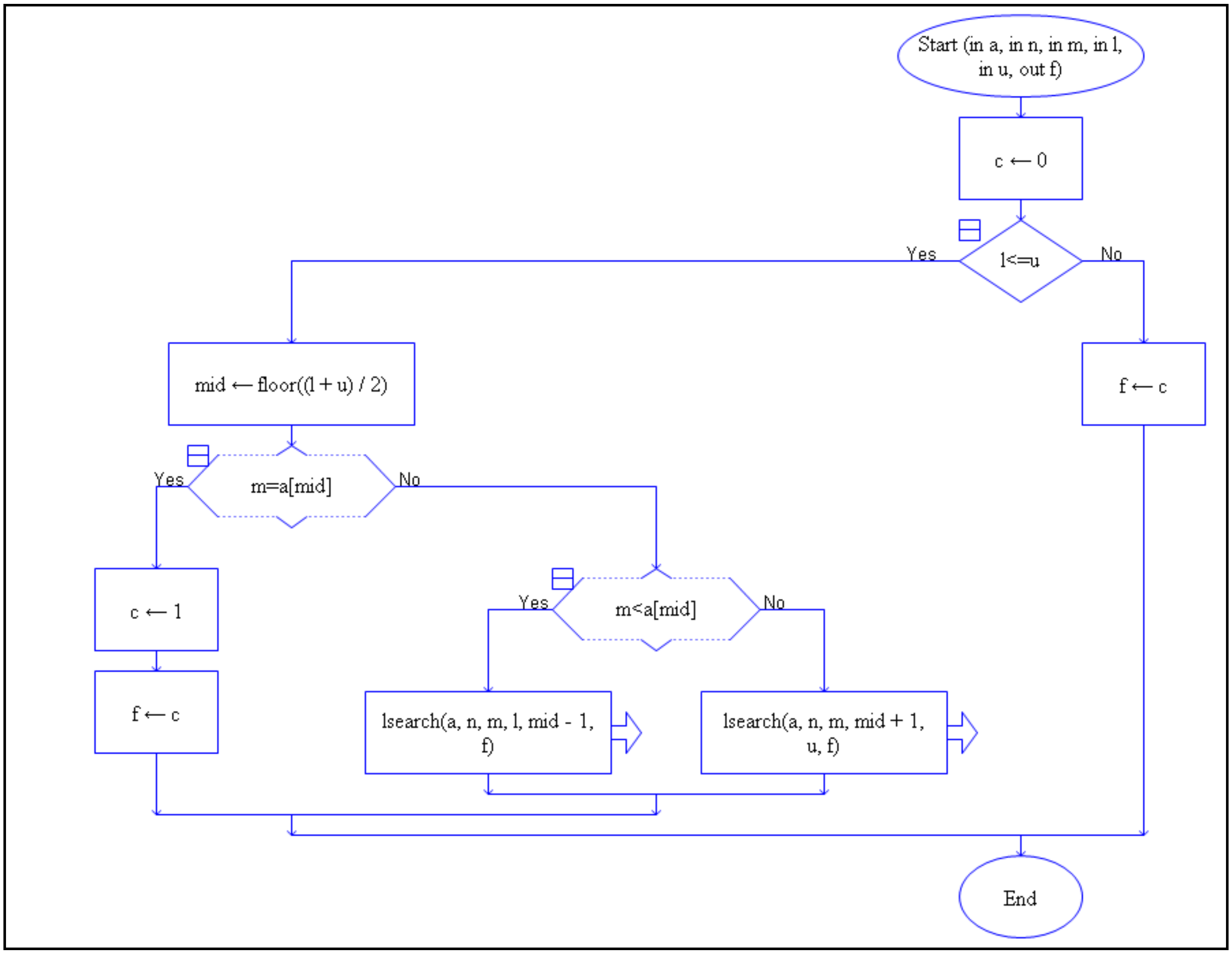
Flow chart for Binary Search



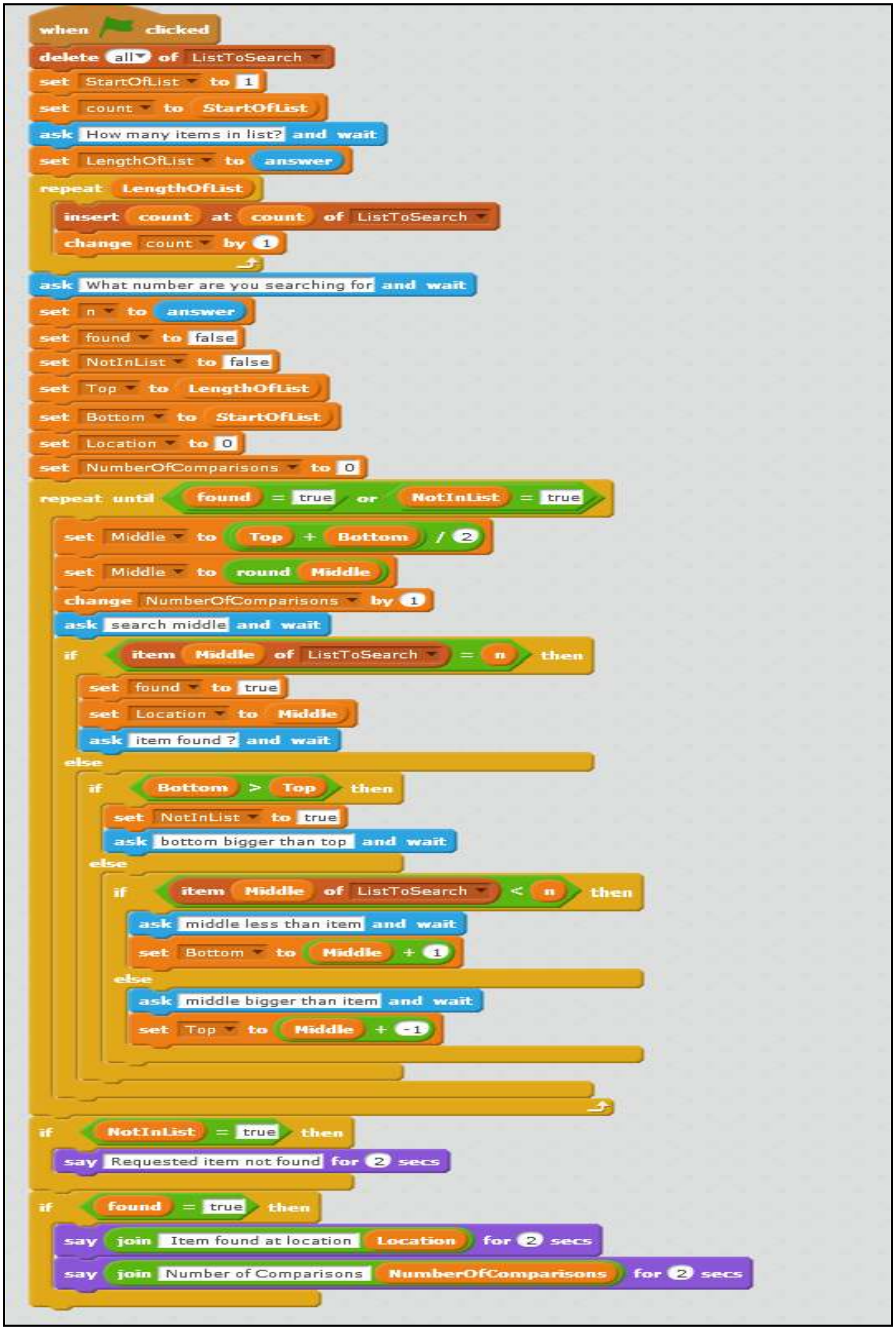
Continuation…



Continuation…


Start (in a, in n, in m, in l, in u, out f)
c ← 0
l<=u
Yes
No
mid ← floor((l + u) / 2)
f ← c
m=a[mid]
Yes
No
c ← 1
f ← c
m<a[mid]
Yes
No
lsearch(a, n, m, l, mid - 1, f)
lsearch(a, n, m, mid + 1, u, f)
End

How to perform Binary Search in Scratch?



```
when green flag clicked
delete all of ListToSearch
set StartOfList to 1
set count to StartOfList
ask How many items in list? and wait
set LengthOfList to answer
repeat LengthOfList
    insert count at count of ListToSearch
    change count by 1
ask What number are you searching for and wait
set n to answer
set found to false
set NotInList to false
set Top to LengthOfList
set Bottom to StartOfList
set Location to 0
set NumberOfComparisons to 0
repeat until found = true or NotInList = true
    set Middle to Top + Bottom / 2
    set Middle to round Middle
    change NumberOfComparisons by 1
    ask search middle and wait
    if item Middle of ListToSearch = n then
        set found to true
        set Location to Middle
        ask item found ? and wait
    else
        if Bottom > Top then
            set NotInList to true
            ask bottom bigger than top and wait
        else
            if item Middle of ListToSearch < n then
                ask middle less than item and wait
                set Bottom to Middle + 1
            else
                ask middle bigger than item and wait
                set Top to Middle + -1
if NotInList = true then
    say Requested item not found for 2 secs
if found = true then
    say join Item found at location Location for 2 secs
    say join Number of Comparisons NumberOfComparisons for 2 secs
```

6.2. Sorting

Sorting is any process of arranging items according to a certain sequence or in different sets, and therefore, it has two common, yet distinct meanings:

1. Ordering: arranging items of the same kind, class or nature, in some ordered sequence,
2. Categorizing: grouping and labeling items with similar properties together (by sorts).

A **sorting algorithm** is an algorithm that puts elements of a list in a certain order. The most-used orders are numerical order and lexicographical order.

Bubble Sort

Bubble sort is a simple **sorting** algorithm that repeatedly steps through the list to be **sorted**, compares each pair of adjacent items and swaps them if they are in the wrong order.

How does it work?

- The bubble sort algorithm sorts the data in the given list.
- It does this by constantly comparing to values that are side-by-side on the list.
- If a value needs to be moved, its value is saved and then moved to the correct location.
- It repeats this until no changes are made to the list.

Example

Let us take the array of numbers "5 1 4 2 8", and sort the array from lowest number to greatest number using bubble sort. In each step, elements written in bold are being compared. Three passes will be required.

First Pass:

(**5 1** 4 2 8) → (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.
(1 **5 4** 2 8) → (1 **4 5** 2 8), Swap since 5 > 4
(1 4 **5 2** 8) → (1 4 **2 5** 8), Swap since 5 > 2
(1 4 2 **5 8**) → (1 4 2 **5 8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

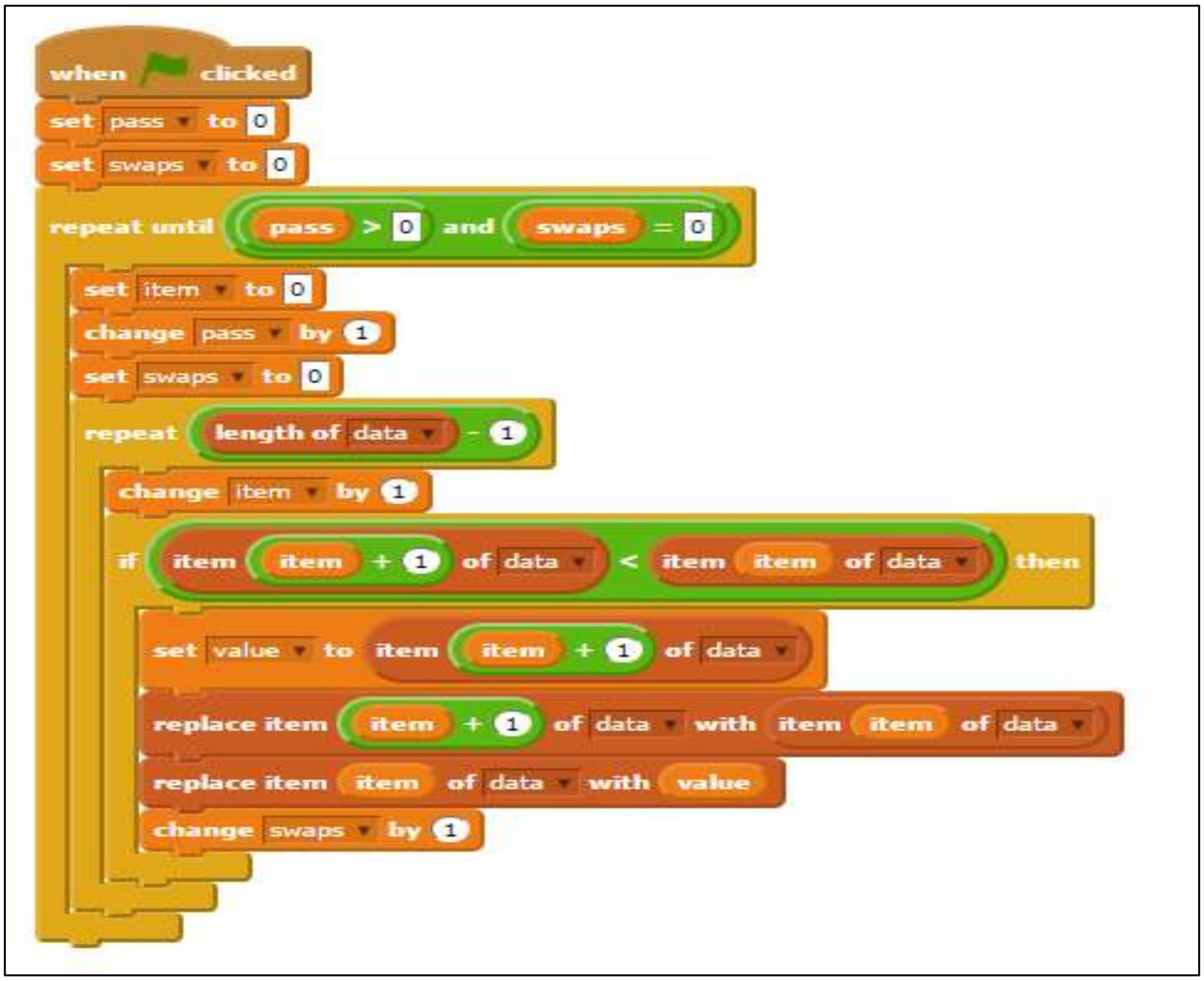
Second Pass:

(**1 4** 2 5 8) → (**1 4** 2 5 8)
(1 **4 2** 5 8) → (1 **2 4** 5 8), Swap since 4 > 2
(1 2 **4 5** 8) → (1 2 **4 5** 8)
(1 2 4 **5 8**) → (1 2 4 **5 8**)
Now, the array is already sorted, but the algorithm does not know if it is completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.

Third Pass:

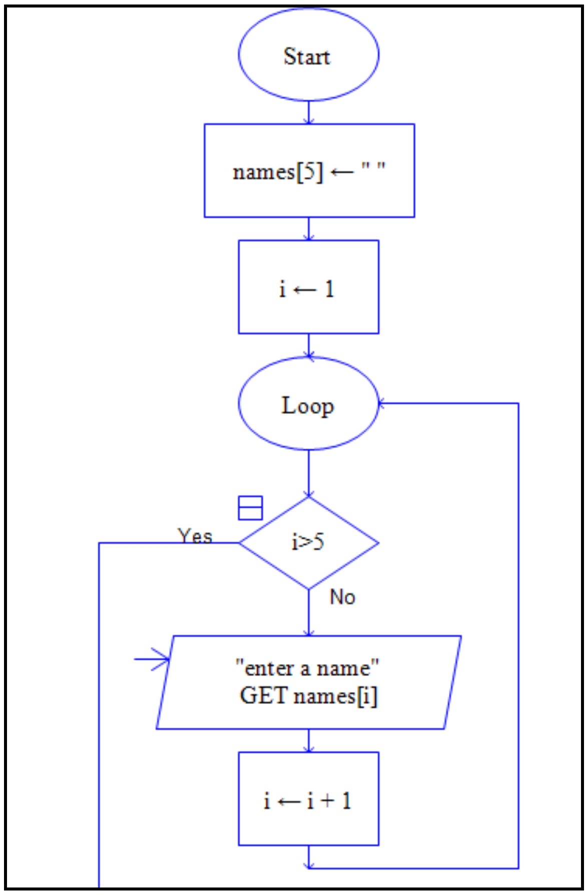
(**1 2** 4 5 8) → (**1 2** 4 5 8)
(1 **2 4** 5 8) → (1 **2 4** 5 8)
(1 2 **4 5** 8) → (1 2 **4 5** 8)
(1 2 4 **5 8**) → (1 2 4 **5 8**)

Bubble Sort in Scratch

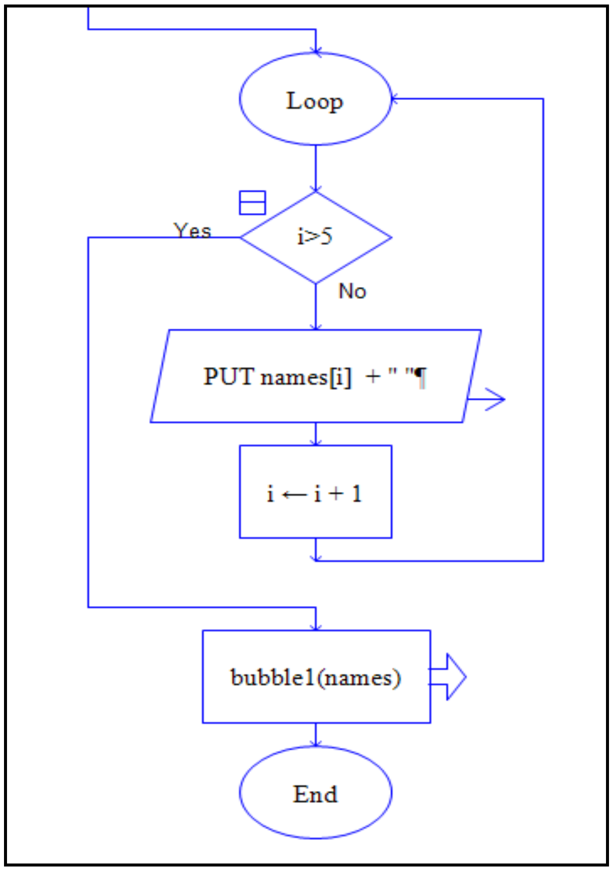


```
when [flag] clicked
set pass to 0
set swaps to 0
repeat until (pass > 0) and (swaps = 0)
    set item to 0
    change pass by 1
    set swaps to 0
    repeat (length of data) - 1
        change item by 1
        if item (item + 1) of data < item (item) of data then
            set value to item (item + 1) of data
            replace item (item + 1) of data with item (item) of data
            replace item (item) of data with value
            change swaps by 1
```

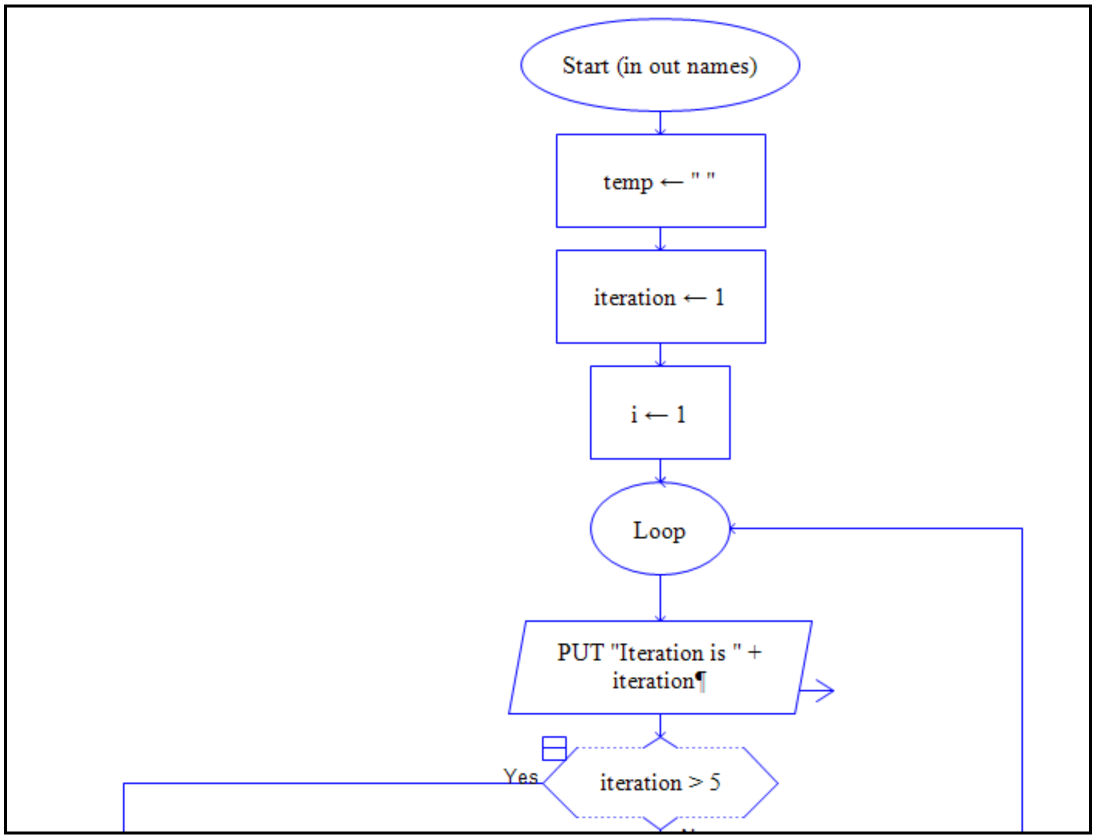
Bubble Sort in RAPTOR



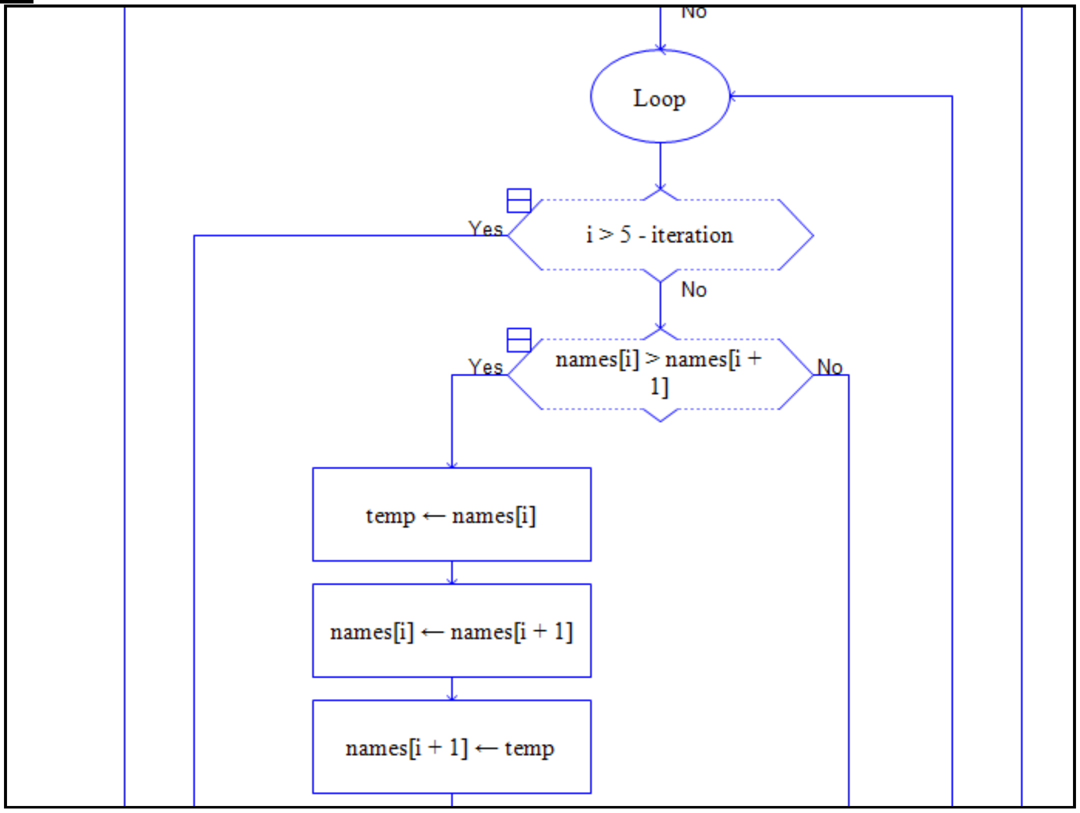
Continuation..



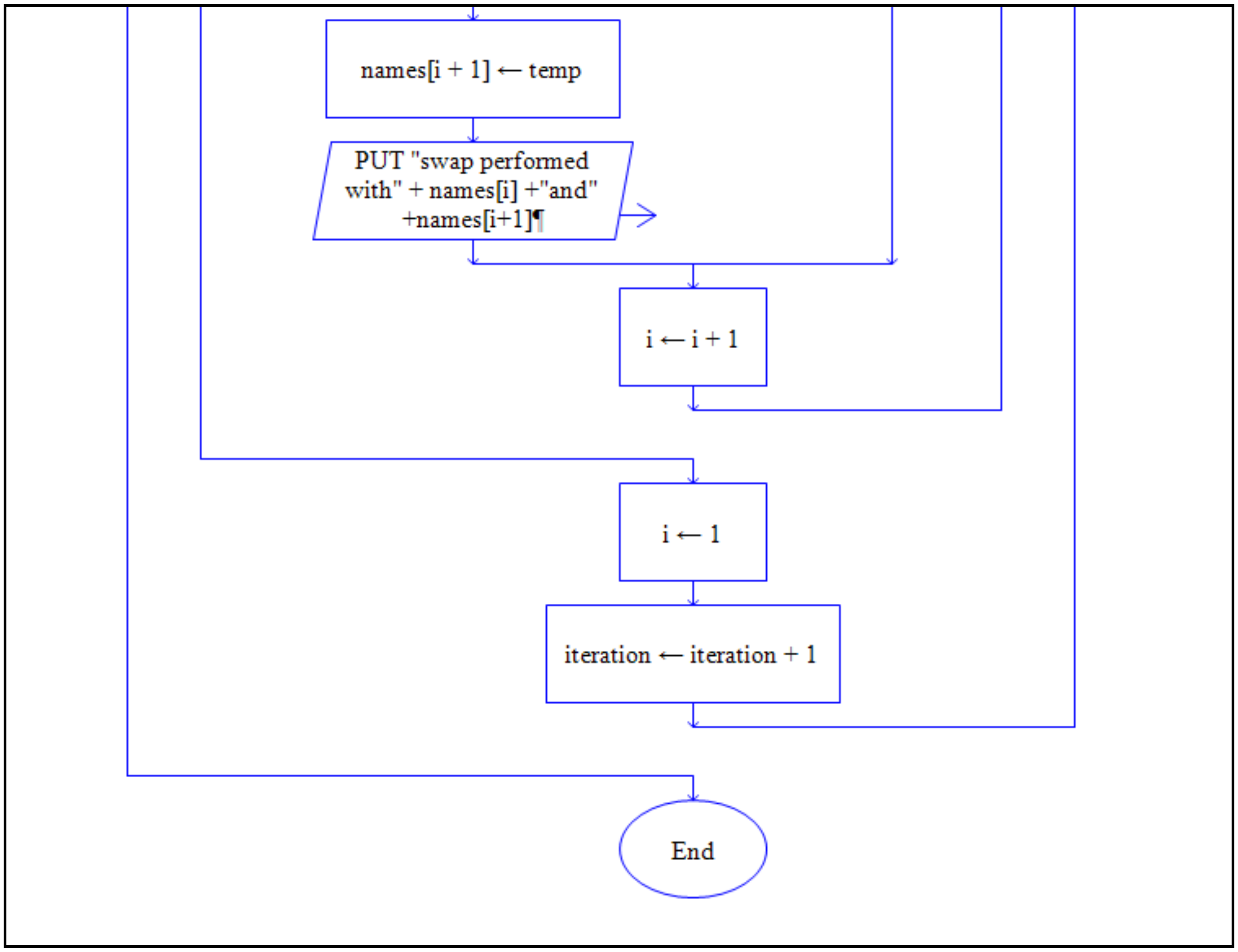
Subprogram for bubble sort:



Continuation…



Continuation…


names[i + 1] ← temp
PUT "swap performed with" + names[i] +"and" +names[i+1]¶
i ← i + 1
i ← 1
iteration ← iteration + 1
End

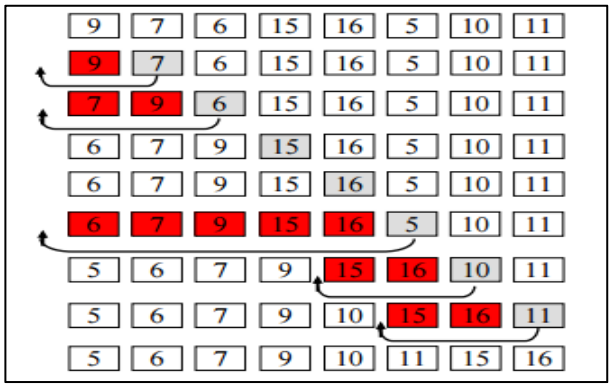
Insertion Sort

Insertion sort is a simple sorting algorithm that builds the final sorted array (or list) one item at a time. **Insertion sort** is an elementary **sorting** algorithm that **sorts** one element at a time. The algorithm takes an element from the list and places it in the correct location in the list.

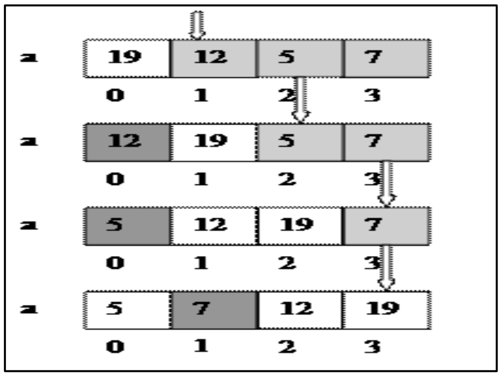
How it does work?

- The bubble sort algorithm sorts the data in the given list.
- The list has 2 parts
- The first part is already sorted and the second part waiting to be sorted
- When the program gets to an unsorted value in a list the program moves the value into its proper location in the first part of the list
- When the program gets the end of the list all of the values have been sorted from least to greatest

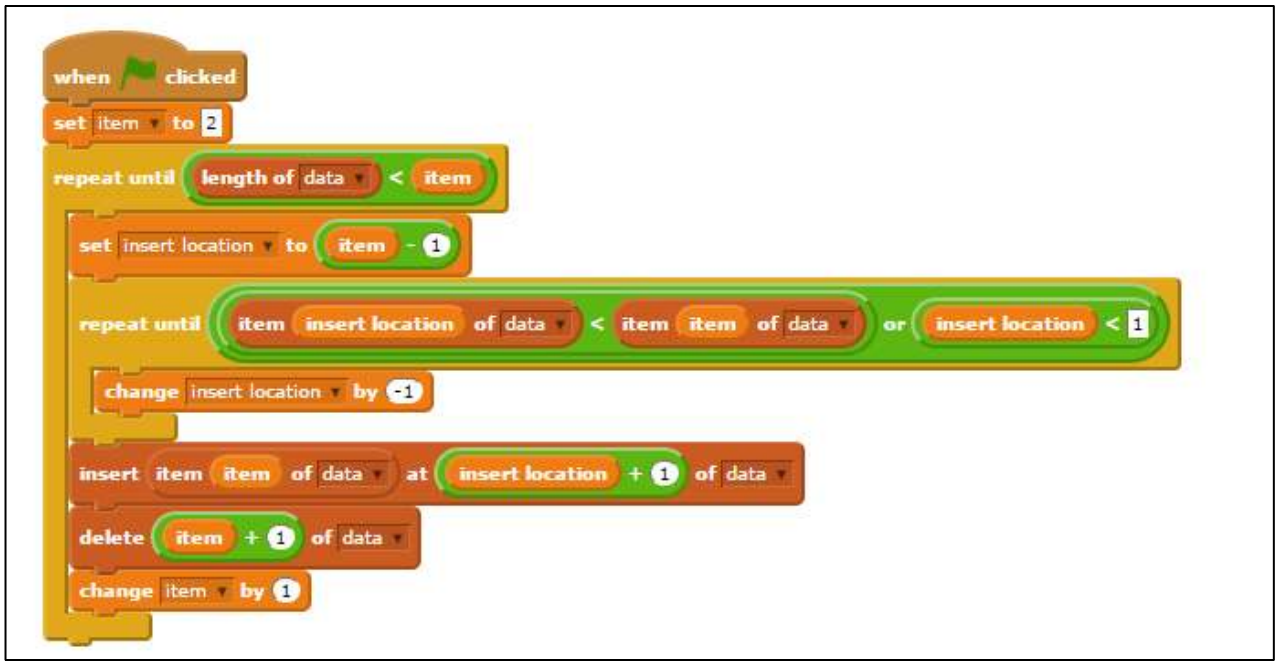
Example-1



Example-2

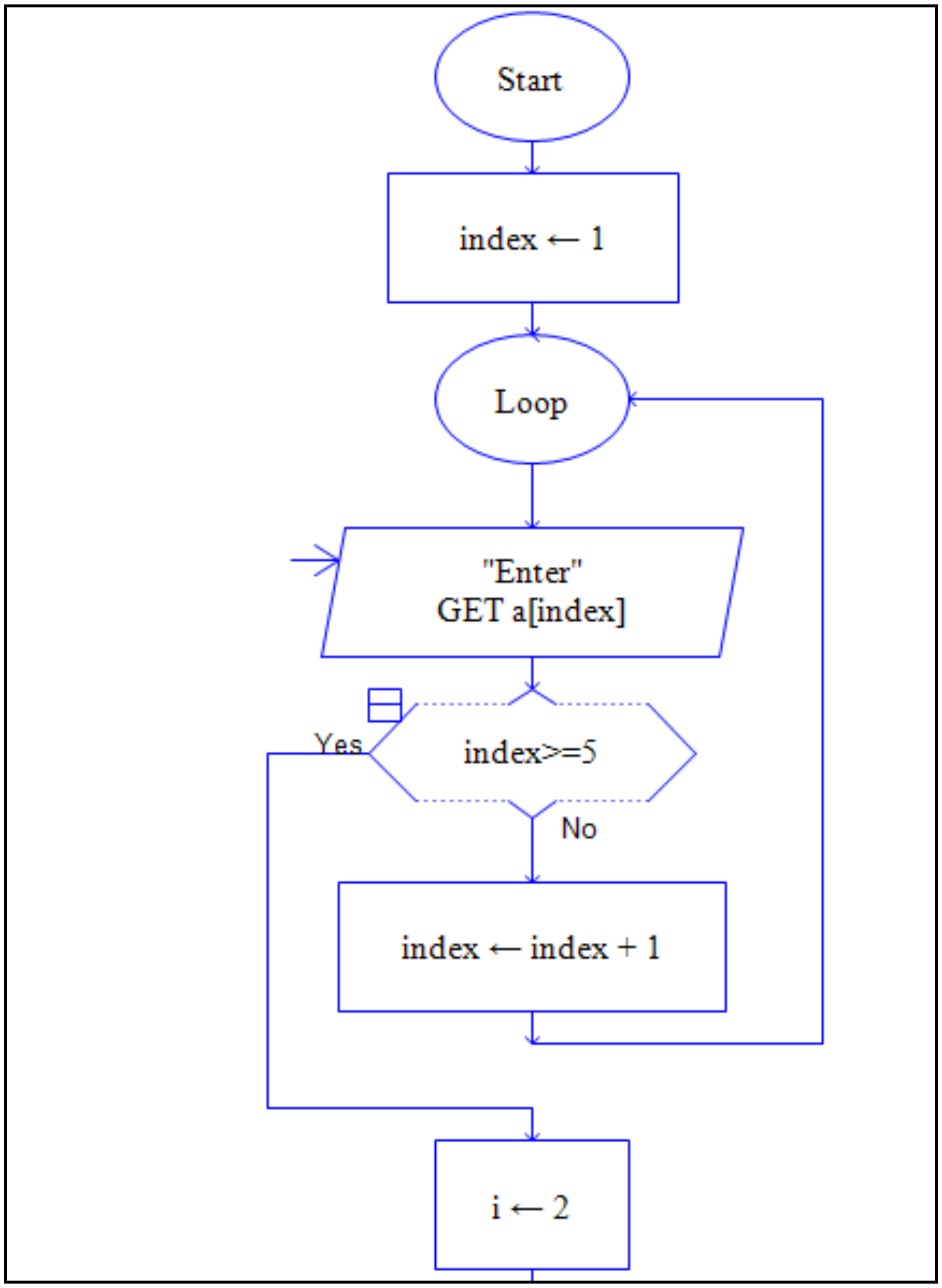


Insertion Sort in Scratch

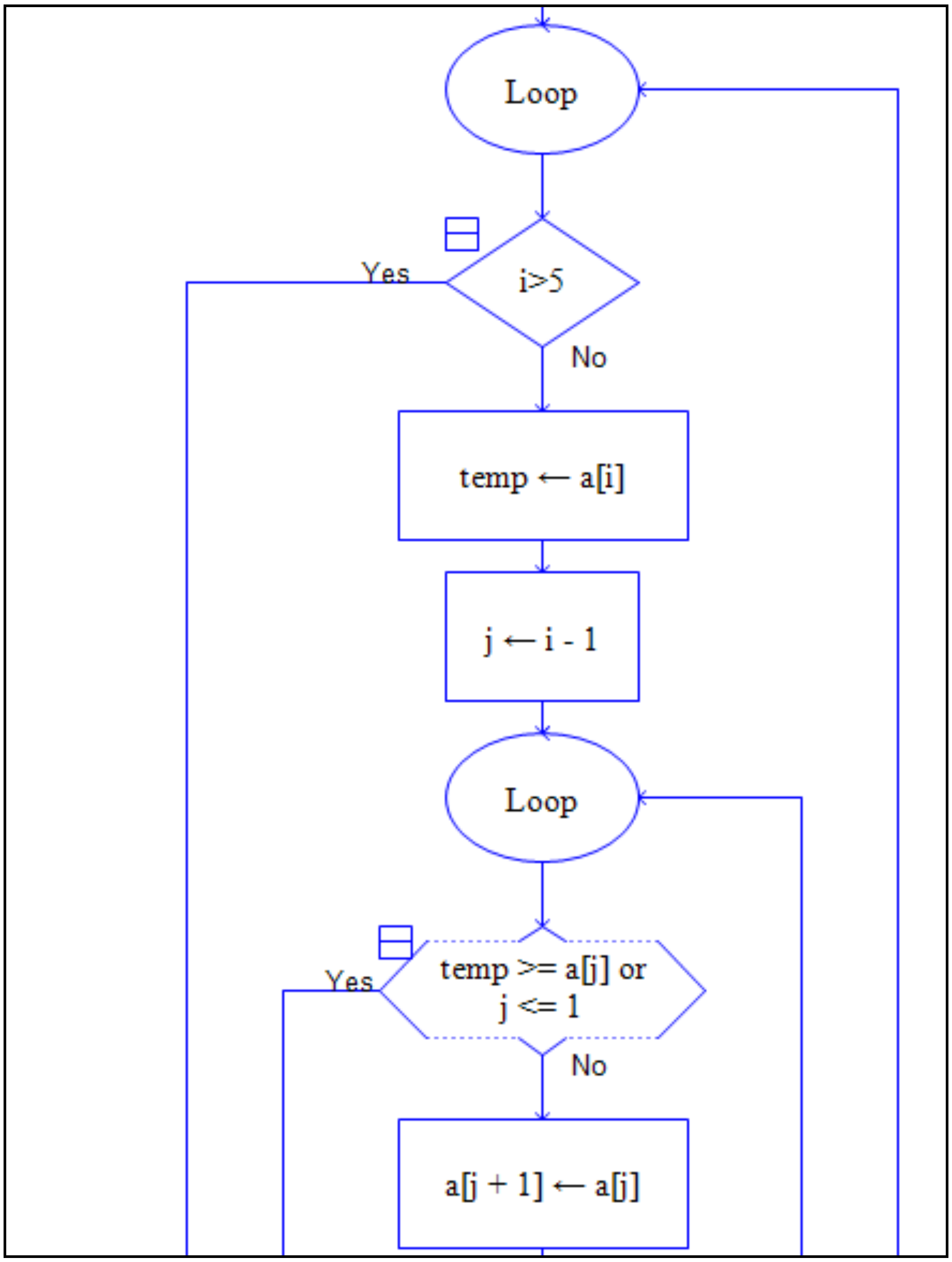


```
when green flag clicked
set item to 2
repeat until < (length of data) < (item) >
    set insert location to ((item) - 1)
    repeat until < < (item (insert location) of data) < (item (item) of data) > or < (insert location) < 1 > >
        change insert location by -1
    end
    insert (item (item) of data) at ((insert location) + 1) of data
    delete ((item) + 1) of data
    change item by 1
end
```

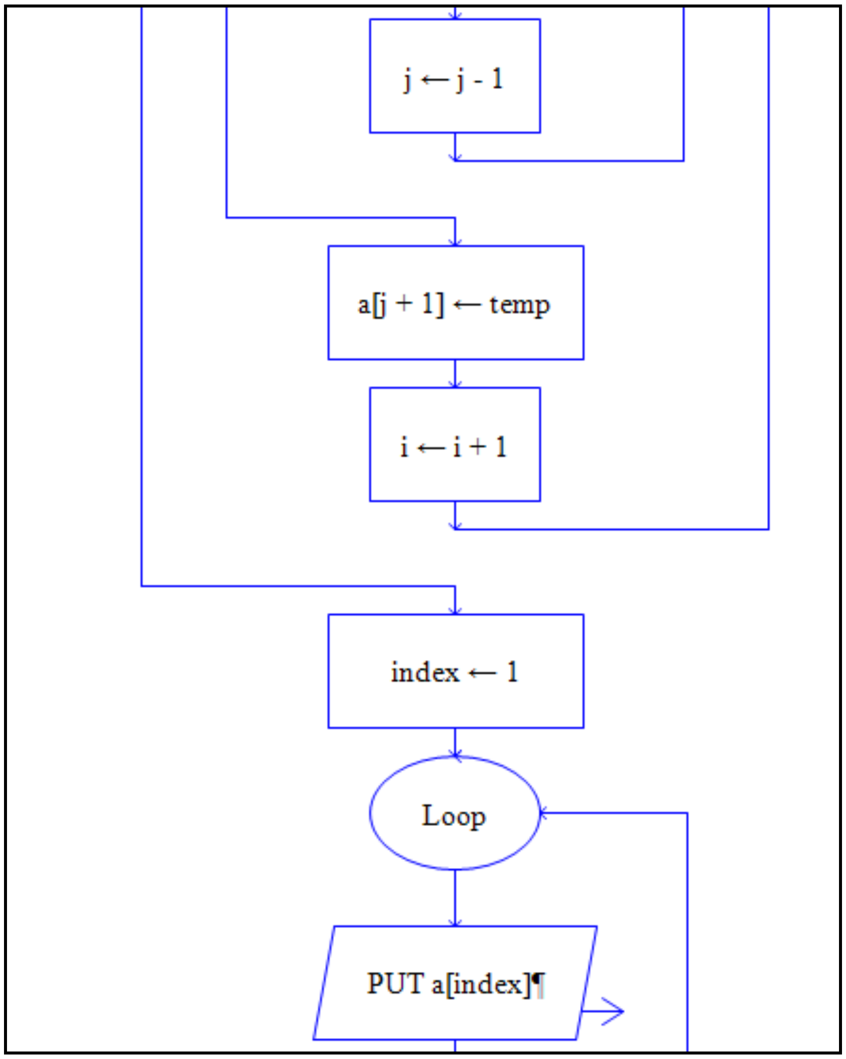
Insertion Sort in RAPTOR

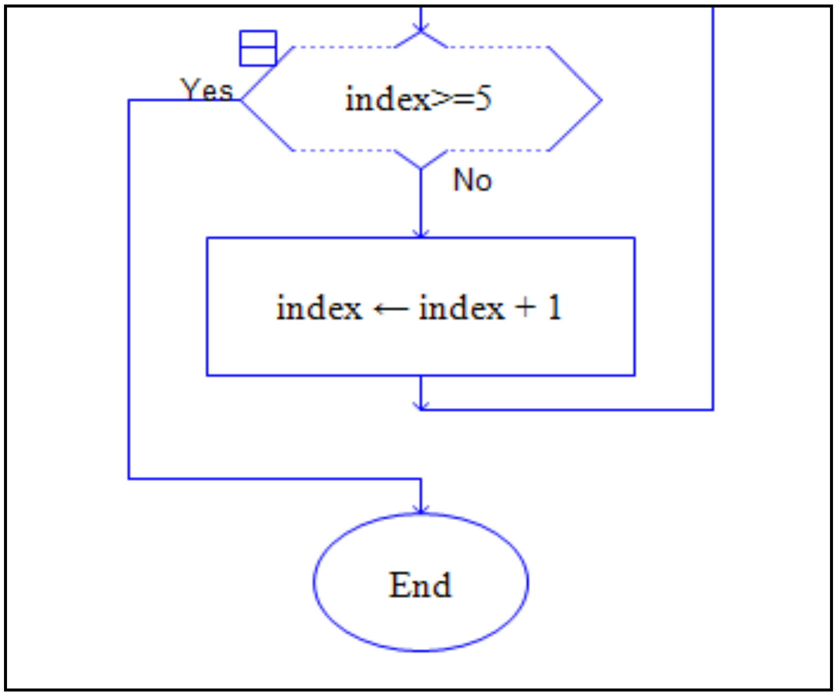


Continuation of Insertion Sort…



Continuation of Insertion Sort…





6.3. Text Processing

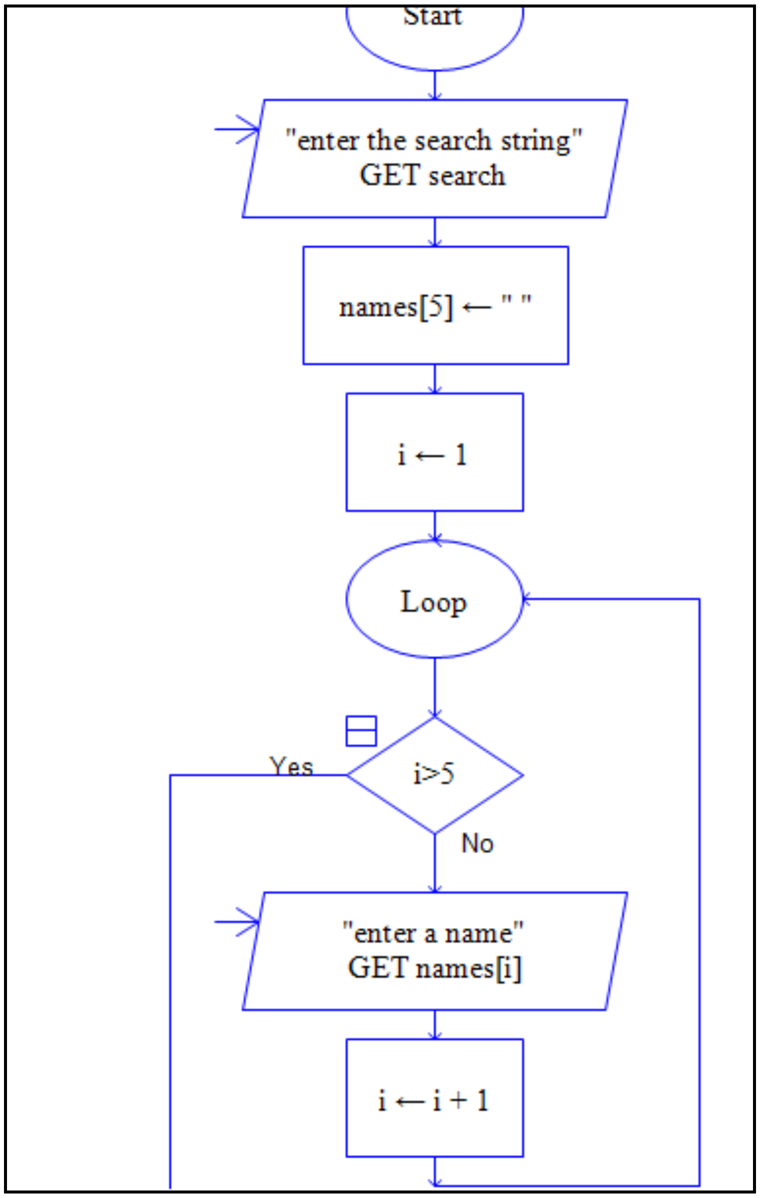
Text searching and its related variations are very important in Computational Thinking. Counting the occurrences of a particular word, searching for a keyword in the text given are the common examples which we touch upon in this section.

String Matching and Occurrence

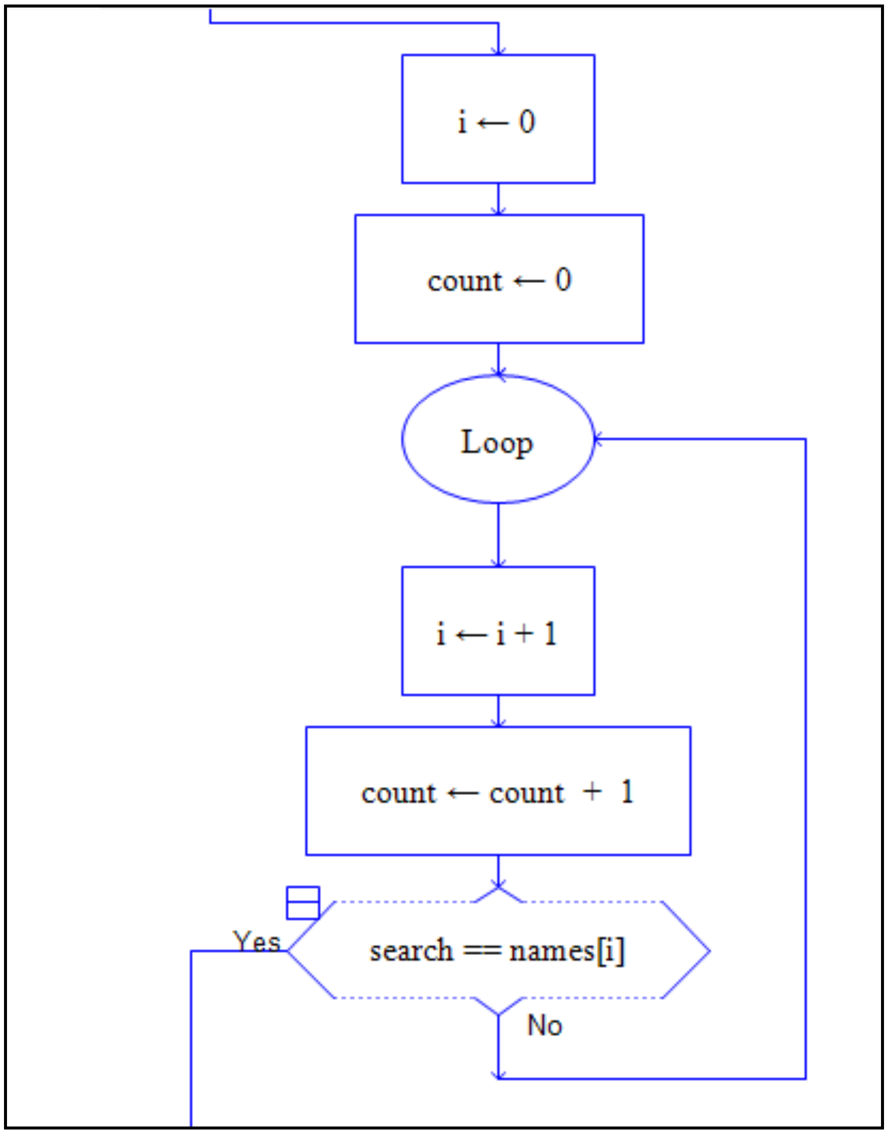
Given a text, to find the no of occurrences of a substring and to search whether the searched substring exists in the text given or not.

Example:
Input a string and check whether the substring exists and to display the count of occurrences of the searched string

Implementation in Raptor



Continuation..



6.4. Exercises

LAQ6.1. Bob secured 34, 25, 42, 27, and 30 marks in Physics, Chemistry, Hindi, Maths, and English respectively. Arrange his marks in ascending order.

LAQ6.2. Scores of batsmen in an IPL match are as follows: 23 runs, 13 runs, 47 runs, 10 runs and 0 run. Sort the scores using insertion sort and bubble sort.

LBQ6.3. A bank was successful in offering Automobile loans at lower rates to increase the business in a particular area and thereby many customers in that region availed the loan. But after a thorough analysis after 1 year, bank realized that most of the customers were irregular in re-payment of automobile loans. The bank maintains the complete information of the customer such as customer loan number and number of monthly repayments. Formulate a strategy to help bank in listing the customer loan numbers in the order of less number of repayments.

LBQ6.4. The Maximum take off weights of various helicopters are given as follows,

Type	MTOW[pounds]
Boeing CH-47D/F Chinook	50,000
AgustaWestland AW101	34,392
Sikorsky CH-53E	73,500
Sikorsky S-92	26,500
Mil Mi-26	123,459
opColumbia Helicopters, Inc 107	19,000
Mil Mi-12	231,485

Search in the data given above for a helicopter with MTOW as 50,000 using suitable search techniques.

LAQ6.5. Take "God can work wonders" as input text and find the no of occurrences of alphabet "o".

LAQ6.6. Given the input "She sells sea shells on the sea shore", find the occurrence of the word "sea" in the input text.

LBQ6.7. You are given two words over the lowercase English alphabet. They have to be combined into a single new word in which all the letters (including repetitions) in the given two words occur in increasing order ('a' is the least and 'z' is the largest) from left to right.

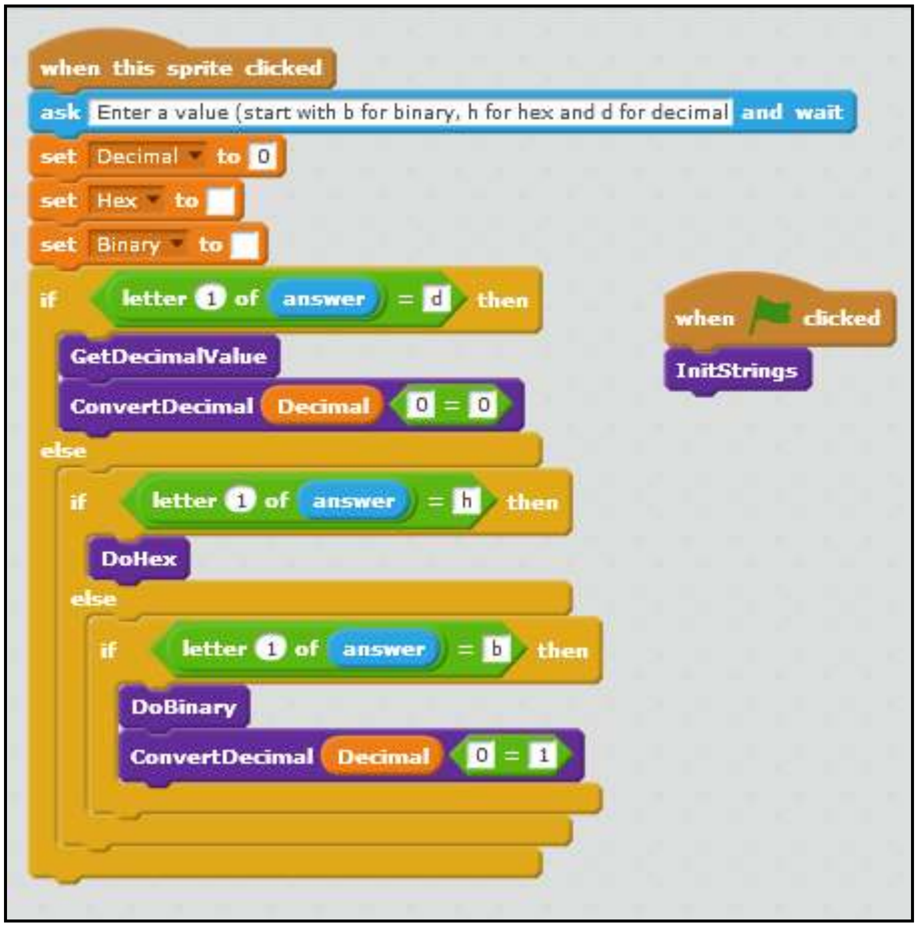
Example: If the two words are "ehllo" and "wrold", the output is "dehllloorw".

Explanation: d, e, h, l, o, r, w is the increasing order among the distinct letters. Since all the letters must occur in the output word, the output is "dehllloorw".

7. PROBLEM ABSTRACTION AND DECOMPOSITION

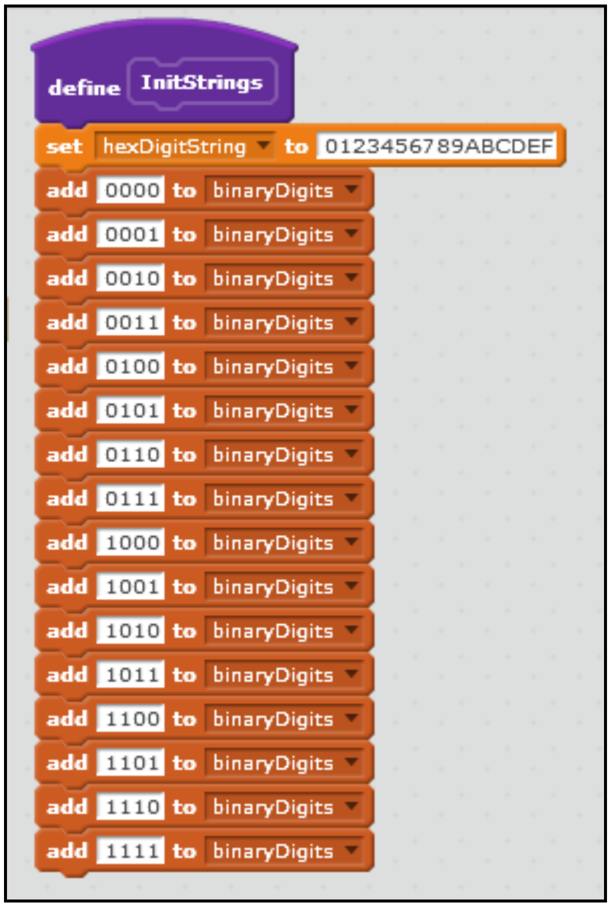
7.1. Modularization

Case study -1: Binary-Decimal-Hexadecimal Conversion



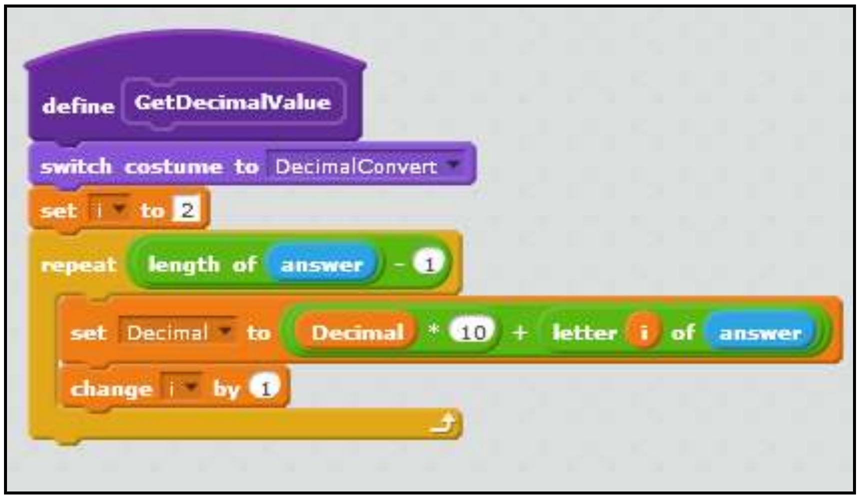
```
when this sprite clicked
ask [Enter a value (start with b for binary, h for hex and d for decimal] and wait
set [Decimal] to [0]
set [Hex] to [ ]
set [Binary] to [ ]
if <(letter (1) of (answer)) = [d]> then
    GetDecimalValue
    ConvertDecimal (Decimal) <[0] = [0]>
else
    if <(letter (1) of (answer)) = [h]> then
        DoHex
    else
        if <(letter (1) of (answer)) = [b]> then
            DoBinary
            ConvertDecimal (Decimal) <[0] = [1]>

when green flag clicked
InitStrings
```

Figure 1: The Main Script Part Where the Modules Have Been Used



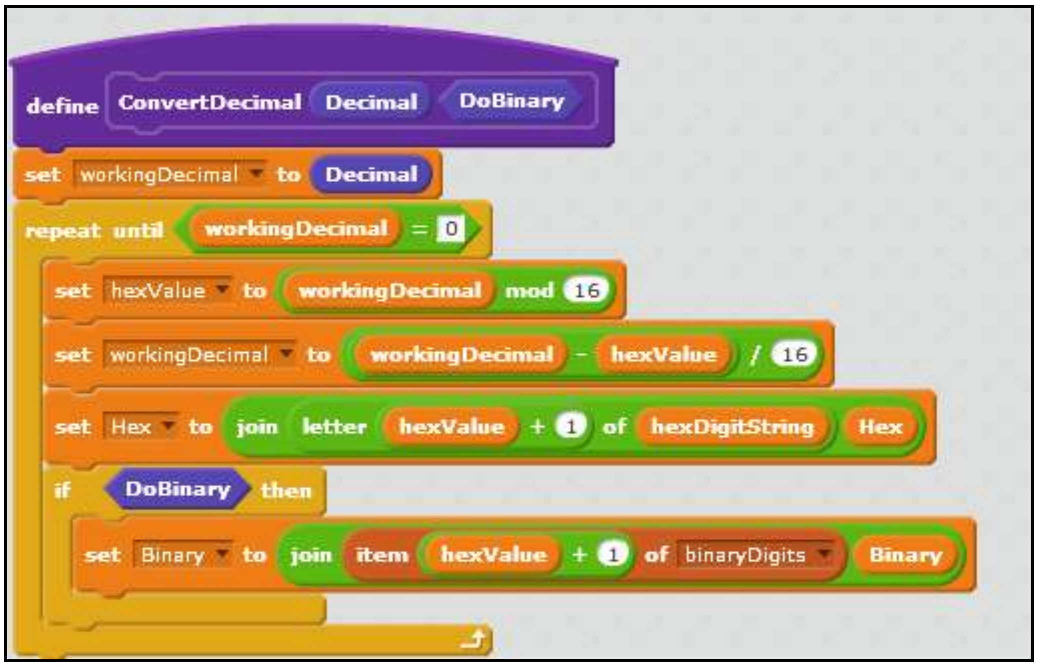
```
define InitStrings
set hexDigitString to 0123456789ABCDEF
add 0000 to binaryDigits
add 0001 to binaryDigits
add 0010 to binaryDigits
add 0011 to binaryDigits
add 0100 to binaryDigits
add 0101 to binaryDigits
add 0110 to binaryDigits
add 0111 to binaryDigits
add 1000 to binaryDigits
add 1001 to binaryDigits
add 1010 to binaryDigits
add 1011 to binaryDigits
add 1100 to binaryDigits
add 1101 to binaryDigits
add 1110 to binaryDigits
add 1111 to binaryDigits
```

Figure 1: Module to Initialize Strings



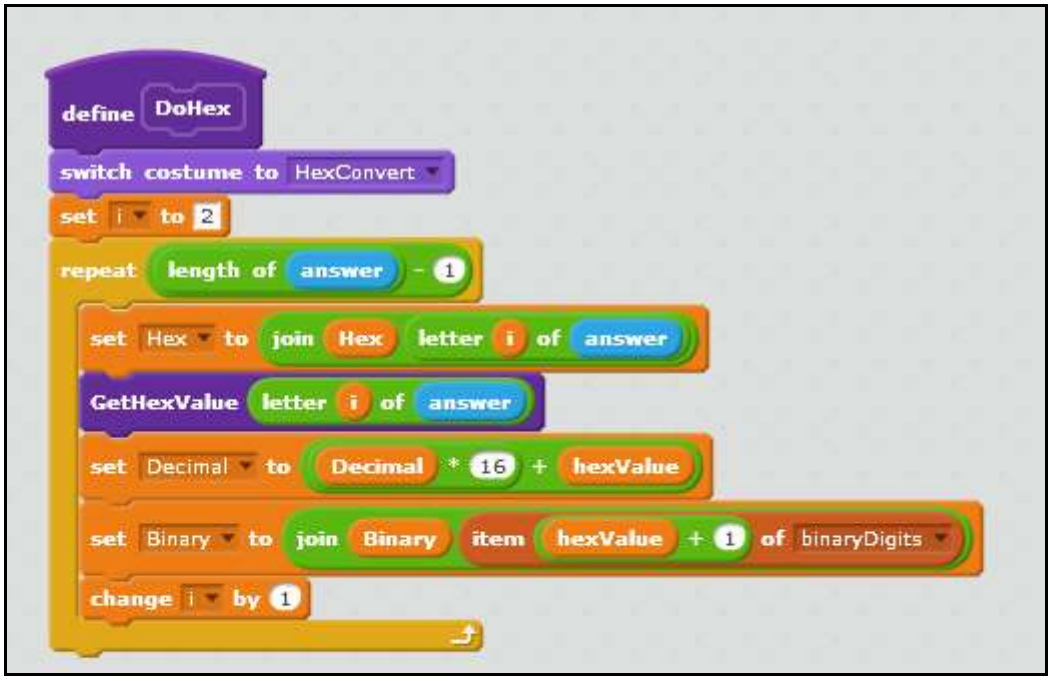
```
define GetDecimalValue
switch costume to DecimalConvert
set i to 2
repeat (length of answer) - 1
    set Decimal to (Decimal * 10) + (letter i of answer)
    change i by 1
```

Figure 3: Module to Get the Decimal Value



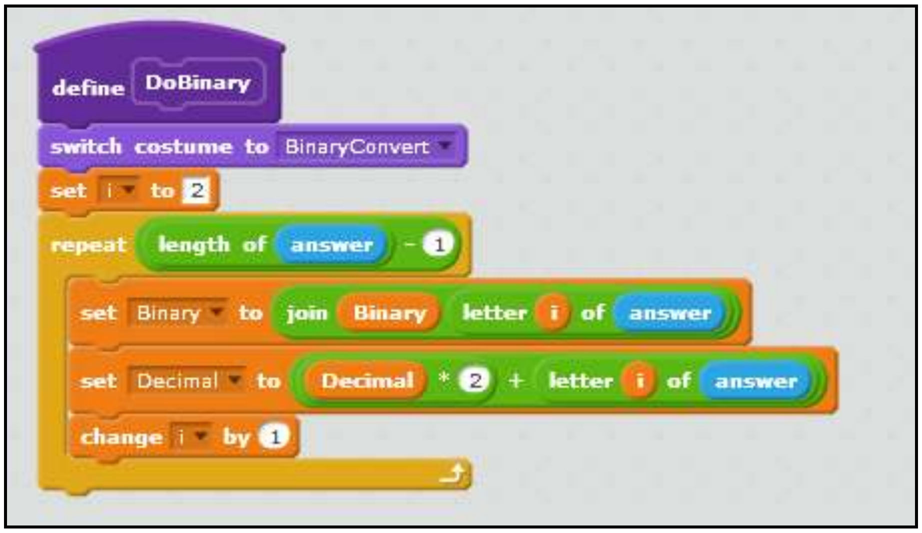
```
define ConvertDecimal (Decimal) <DoBinary>
set workingDecimal to (Decimal)
repeat until <(workingDecimal) = 0>
    set hexValue to ((workingDecimal) mod 16)
    set workingDecimal to (((workingDecimal) - (hexValue)) / 16)
    set Hex to (join (letter ((hexValue) + 1) of (hexDigitString)) (Hex))
    if <DoBinary> then
        set Binary to (join (item ((hexValue) + 1) of binaryDigits) (Binary))
```

Figure 4: Module to Convert Decimal Value to Binary Value



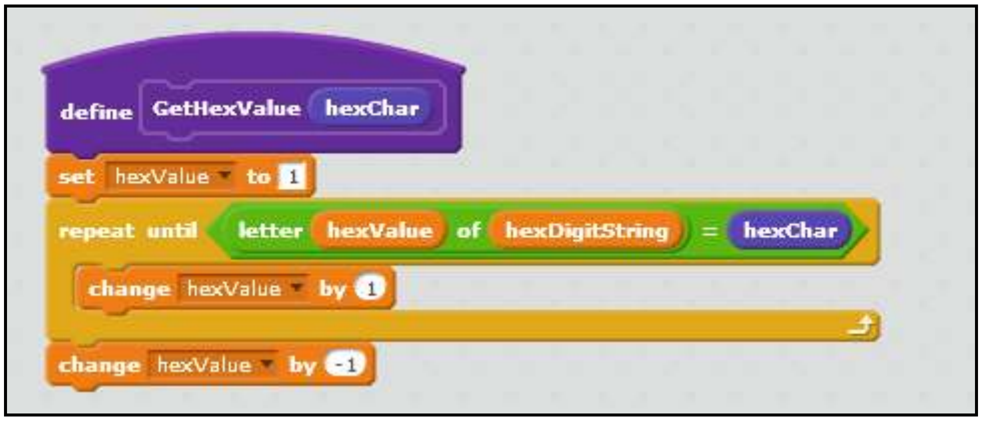
```
define DoHex
switch costume to HexConvert
set i to 2
repeat ((length of (answer)) - 1)
    set Hex to (join (Hex) (letter (i) of (answer)))
    GetHexValue (letter (i) of (answer))
    set Decimal to (((Decimal) * 16) + (hexValue))
    set Binary to (join (Binary) (item ((hexValue) + 1) of binaryDigits))
    change i by 1
```

Figure 5: Module for Converting to Hexadecimal Value



```
define DoBinary
switch costume to BinaryConvert
set i to 2
repeat (length of answer) - 1
    set Binary to join Binary letter i of answer
    set Decimal to (Decimal * 2) + letter i of answer
    change i by 1
```

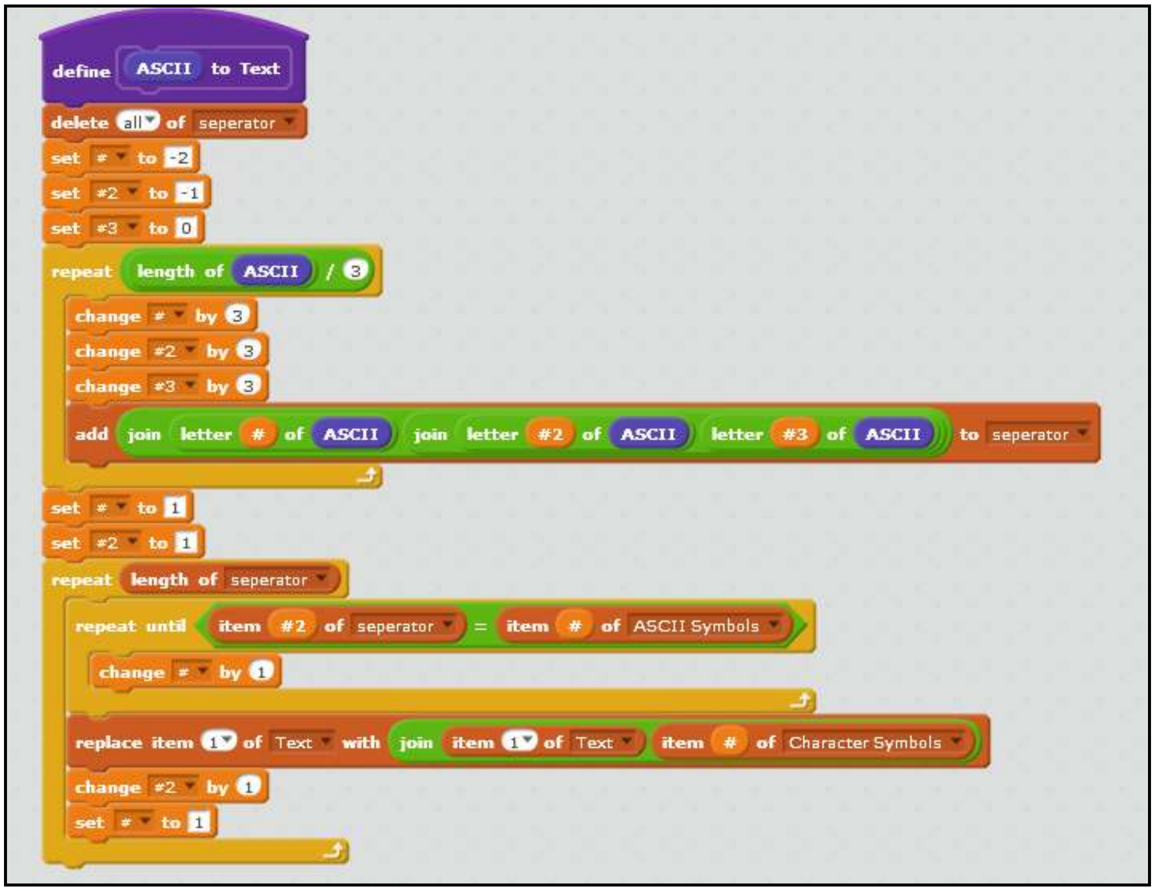
Figure 6: Module for Converting to Binary Value



```
define GetHexValue hexChar
set hexValue to 1
repeat until letter hexValue of hexDigitString = hexChar
    change hexValue by 1
change hexValue by -1
```

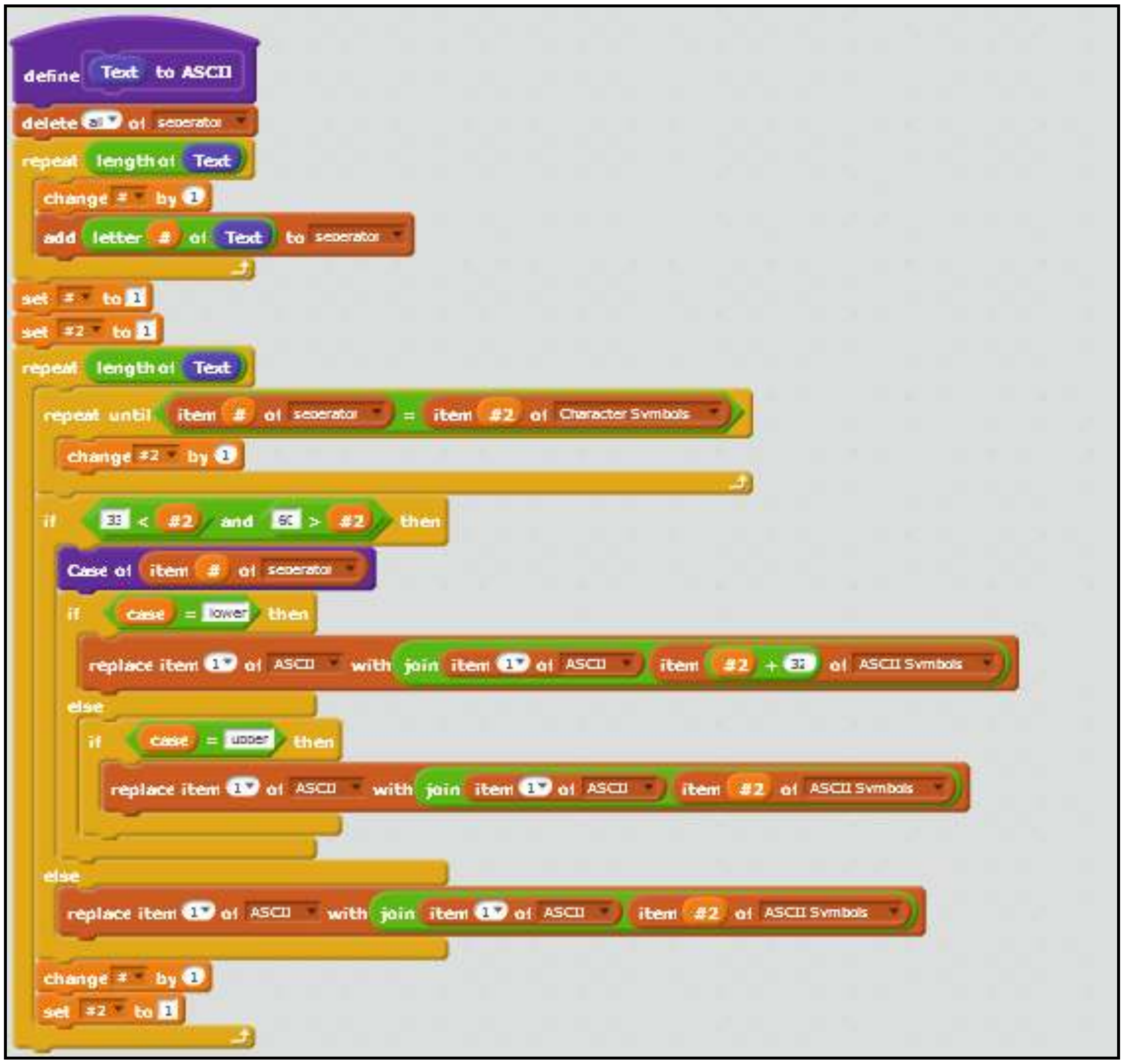
Figure 7: Module for Getting the Hexadecimal Value

Case study -2: Conversion of ASCII to Text and Text to ASCII



```
define ASCII to Text
delete all of seperator
set # to -2
set #2 to -1
set #3 to 0
repeat (length of ASCII / 3)
    change # by 3
    change #2 by 3
    change #3 by 3
    add join letter # of ASCII join letter #2 of ASCII letter #3 of ASCII to seperator
set # to 1
set #2 to 1
repeat length of seperator
    repeat until item #2 of seperator = item # of ASCII Symbols
        change # by 1
    replace item 1 of Text with join item 1 of Text item # of Character Symbols
    change #2 by 1
    set # to 1
```

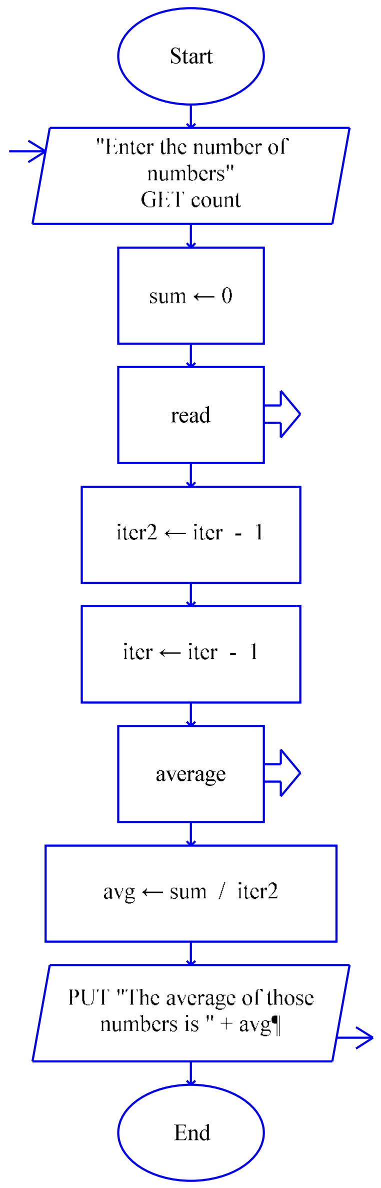
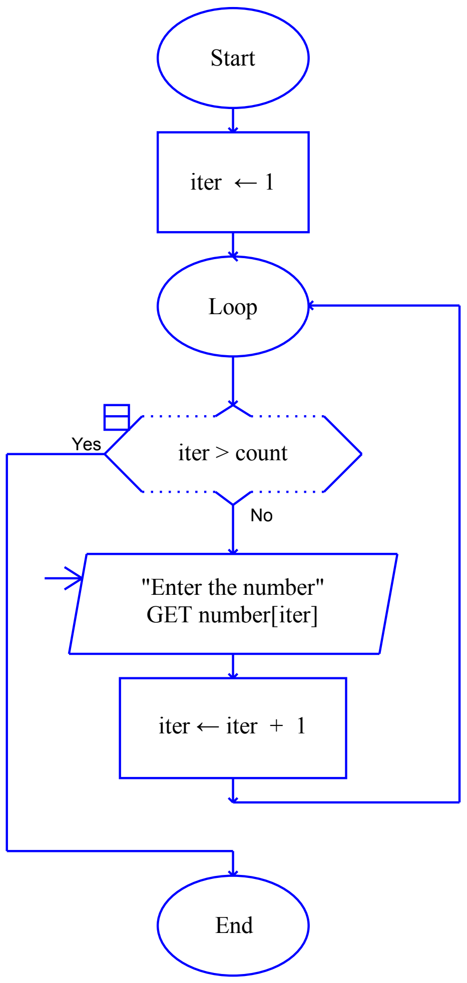
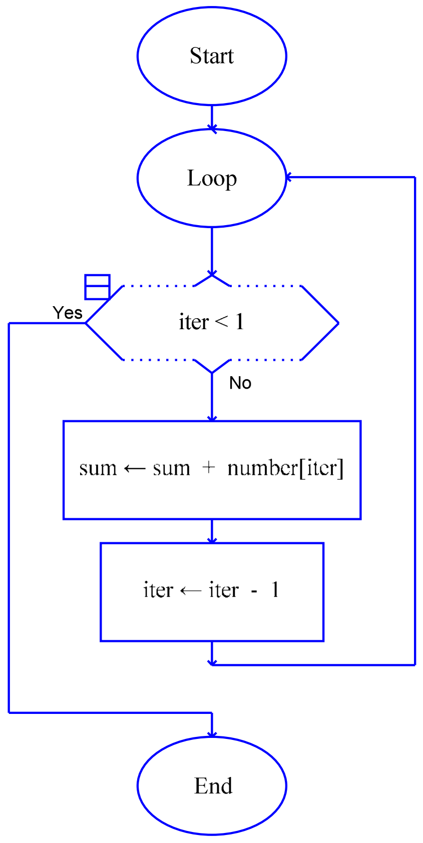
Figure 8: Module for Converting ASCII to Text



```
define (Text) to ASCII
delete all of seperator
repeat (length of (Text))
    change # by 1
    add (letter (#) of (Text)) to seperator
set # to 1
set #2 to 1
repeat (length of (Text))
    repeat until <(item (#) of seperator) = (item (#2) of Character Symbols)>
        change #2 by 1
    if <<33 < (#2)> and <60 > (#2)>> then
        Case of (item (#) of seperator)
        if <(case) = lower> then
            replace item 1 of ASCII with (join (item 1 of ASCII) (item ((#2) + 32) of ASCII Symbols))
        else
            if <(case) = upper> then
                replace item 1 of ASCII with (join (item 1 of ASCII) (item (#2) of ASCII Symbols))
    else
        replace item 1 of ASCII with (join (item 1 of ASCII) (item (#2) of ASCII Symbols))
    change # by 1
    set #2 to 1
```

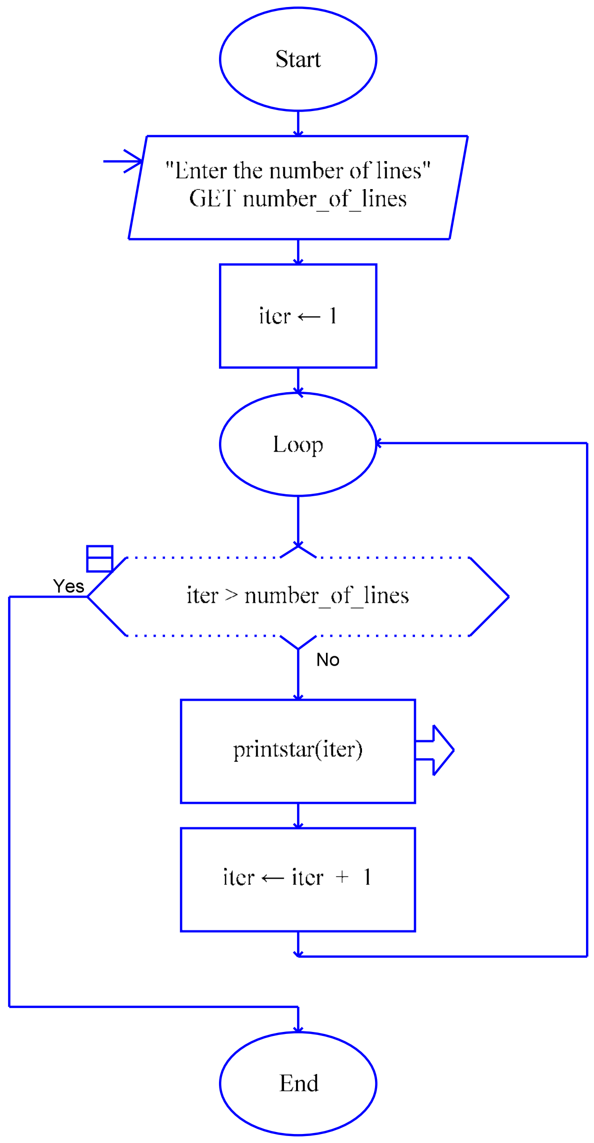
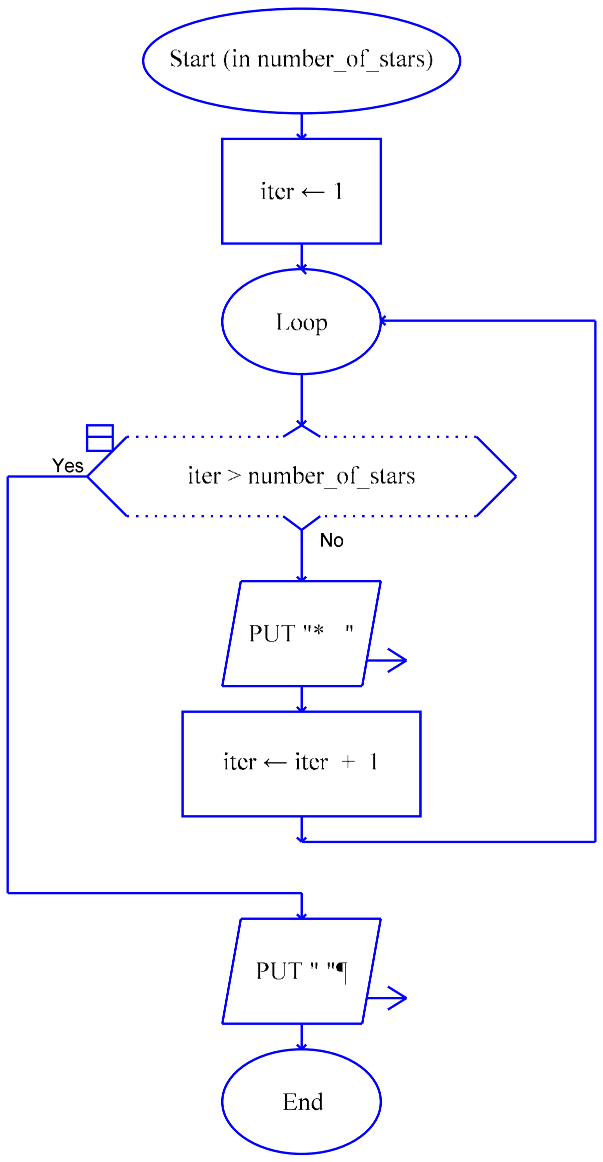
Figure 9: The Module for Converting Text to ASCII

Case study -3: Finding the average of a set of numbers read from the user, using two sub-charts namely “read” and “average”.

main	**read**	**average**
 Start "Enter the number of numbers" GET count sum ← 0 read iter2 ← iter - 1 iter ← iter - 1 average avg ← sum / iter2 PUT "The average of those numbers is " + avg¶ End	 Start iter ← 1 Loop Yes iter > count No "Enter the number" GET number[iter] iter ← iter + 1 End	 Start Loop Yes iter < 1 No sum ← sum + number[iter] iter ← iter - 1 End

Case study - 4: Flowchart to read a number (number of lines) as input print the following pattern using a function by the name of "printstar". If "number_of_lines" is equal to 4, the pattern would be as follows.

```
*
* *
* * *
* * * *
```

main	printstar
 Start "Enter the number of lines" GET number_of_lines iter ← 1 Loop Yes iter > number_of_lines No printstar(iter) iter ← iter + 1 End	 Start (in number_of_stars) iter ← 1 Loop Yes iter > number_of_stars No PUT "* " iter ← iter + 1 PUT " "¶ End

7.2. Exercises

LAQ7.1. Using the concept of modularization in Scratch programming, implement addition and multiplication tables.

LBQ7.2. Write a Scratch program for managing issue and return of books in a college library.

LAQ7.3. Implement matrix addition and multiplication using modularization technique in Raptor.

LBQ7.4. Draw a Raptor flowchart interpreter for the order and billing of an inventory system.

CREDITS

1. *http://www.howtogeek.com/school/microsoft-excel-formulas-and-functions*
2. *http://www.experiglot.com/2007/11/19/how-to-use-vlookup-in-excel-a-simple-tutorial-part-i/?s=hlookup&Submit=Go%21*
3. *http://best-excel-tutorial.com/58-excel-functions/112-hlookup*
4. *https://support.office.com/en-us/article/Excel-functions-by-category-7fd9655a-4e87-400a-ae5c-c48f16afde0c*
5. *https://www.ablebits.com/office-addins-blog/2012/05/03/tutorial-excel-macros/#create-macros*
6. *http://scratch.mit.edu*
7. *http://raptor.martincarlisle.com*

Lab Manual Preparation Team

Dept. of CSE,
Amrita School of Engineering,
Coimbatore.

Course Mentor: Ms. C.K. Shyamala

Team Coordination: Dr. M. Senthilkumar

Tools Exploration: Mr. K. Raghesh Krishnan, Mr. V. Dayanand and Ms. V. Gayathri

Contributors: Ms.Dhanya N M, Ms.Sini Raj P, Ms. K.R.Bindu, Dr.B.Rajathilagam, Mr.S.Thangavelu, Dr.D.Venkataraman, Dr.C.Shunmuga Velayutham, Ms.K.Nalinadevi and Dr.Vidhya Balasubramanian.

Record your observations…