4 Programming

This chapter will cover the following:

- Analyzing problems
- Control structures
- Providing solutions to day-to-day problems
- Developing programs using *sequence* and *selection* control structures
- Applications of mobile and smart devices.

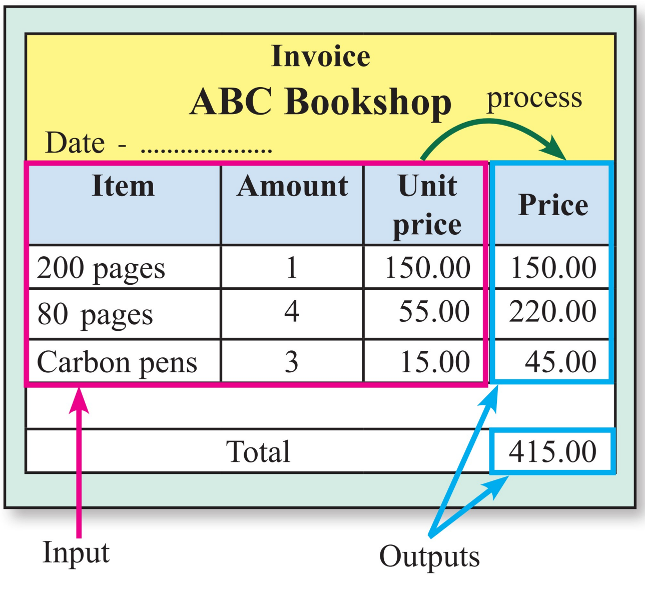
4.1 Problem analysis

Analyzing a problem involves dividing the problem into smaller segments of the problem and examine. This will make it easier to solve the problem.

For example, let us consider an invoice issued by a stationery shop.

To calculate the price (amount) of each item, number of items and unit price are required. The items required to prepare the invoice, are called **input**. Calculating the total price for each item and the value of total bill is known as **processing**. Price (amount) for each item and total bill value are known as **output**.

Hence, let us analyze the above bill and identify input, process and output.



Invoice

ABC Bookshop

Date - ....

Item	Amount	Unit price	Price
200 pages	1	150.00	150.00
80 pages	4	55.00	220.00
Carbon pens	3	15.00	45.00
Total			415.00

Input :	Item description, number of items, unit price
Process :	For an item purchased; Amount = number of items × unit price Total bill = total price (amount) of all items purchased
Output :	Amount to be paid

In order to develop a computer program, it is essential to identify the *inputs, process and outputs* by analyzing a problem. (see figure 4.1)

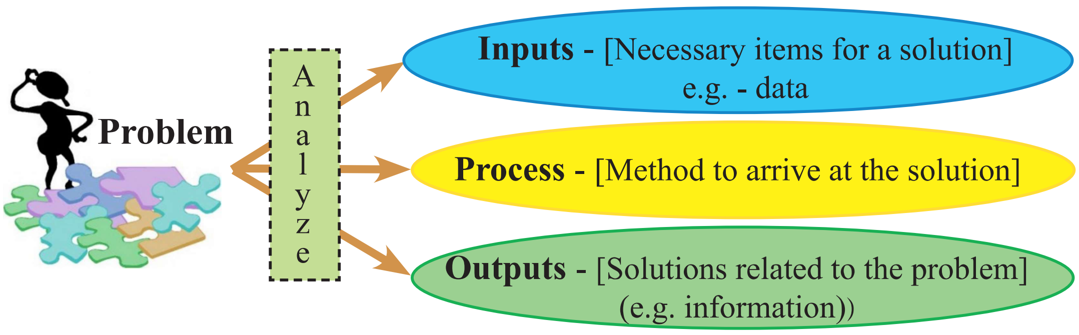

Problem
Analyze
Inputs - [Necessary items for a solution] e.g. - data
Process - [Method to arrive at the solution]
Outputs - [Solutions related to the problem] (e.g. information))

Figure 4.1 : Problem Analysis

Note - The problem needs analyzing before developing computer programs.

When analyze a problem, it is possible to identify *inputs, process and outputs*.

Example 1

Problem :	Find the year of birth when a person's National Identity Card number is given.
Input :	National Identity Card number
Process :	Select the first two digits in the identity card number
Output :	Year of birth

987654321V

Example 2

Problem :	Find the cost of purchasing five pens
Input :	Price of a pen
Process :	Calculating cost (total = price of a pen × 5)
Output :	Total amount

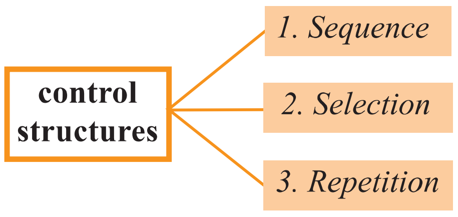
Figure 4.2 : Pens

Refer to workbook for Activity 4.1.

4.2 Control structures

A control structure is a block in a program that analyses variables and chooses a direction of flow of the program.

You have learnt in chapter five of Grade 7 book that there are three types of control structures as *sequence, selection* and *repetition.*

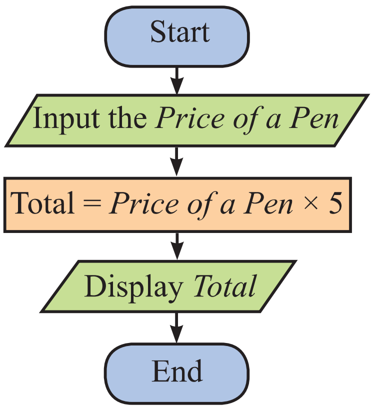


Note - In inputs and outputs are denoted by the symbol ▱ and the processes are by ▭ symbol.

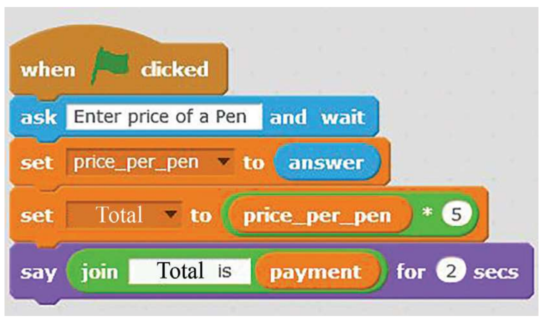
4.2.1 Sequence

If the steps are carried out one after the other from the beginning to the end in a particular consecutive order, it is called a *sequence.*

The flowchart and the Scratch program below is equivalent to the Example 2 in page 40.



Flowchart 1 : finding cost of five pens



```
when [flag] clicked
ask [Enter price of a Pen] and wait
set [price_per_pen] to (answer)
set [Total] to ((price_per_pen) * (5))
say (join [Total is] (payment)) for (2) secs
```

Scratch program 1: find the cost of 5 pens when the unit price is provided

The price of the pen is shown by the *price_per_pen* variable and the amount to be paid is shown by the *Total* variable.

4.2.2 Selection

Selection decides which step(s) are executed depending on whether a condition of an algorithm is satisfied or not.

For example, consider a rainy day. If it rains students are asked to go to the library. If it does not rain, students are asked to go to the playground.

The decision box in flowcharts, is used to show the selection control structure (See Figure 4.3). If the condition is true, it is directed towards "Yes". If it is false, it is directed towards "No". The following symbol is used To indicate the decision making;

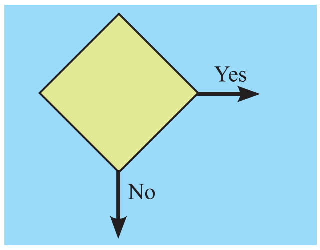


Figure 4.3 : Control structure

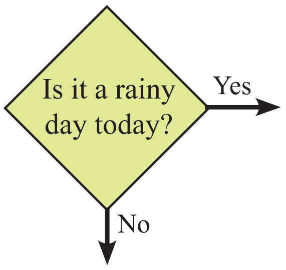
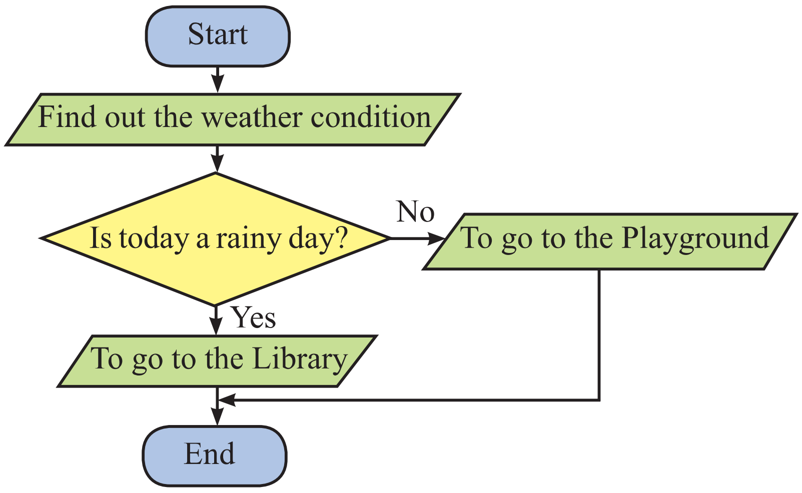


Figure 4.4 : Decision on whether it is a rainy day or not

Example 1 Indicate the above example in a flowchart.



Flow chart 2 : Going to the playground or the library according to the weather condition

Example 2 Making decisions when playing *Snakes and Ladders* Game

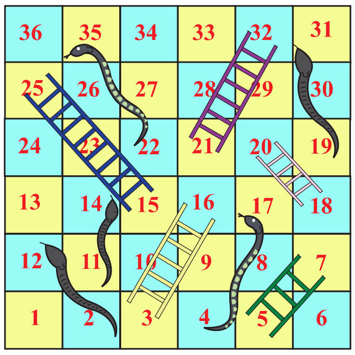
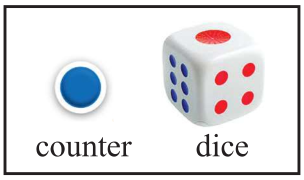


Figure 4.5 : Snakes and Ladders Board

Snakes and Ladders is a popular game that can be played by an individual or by a group of players. In this game, there are number of boxes from beginning (1) to end (36). Each *ladder* and *snake* has two boxes connected at ends (Refer figure 4.5).

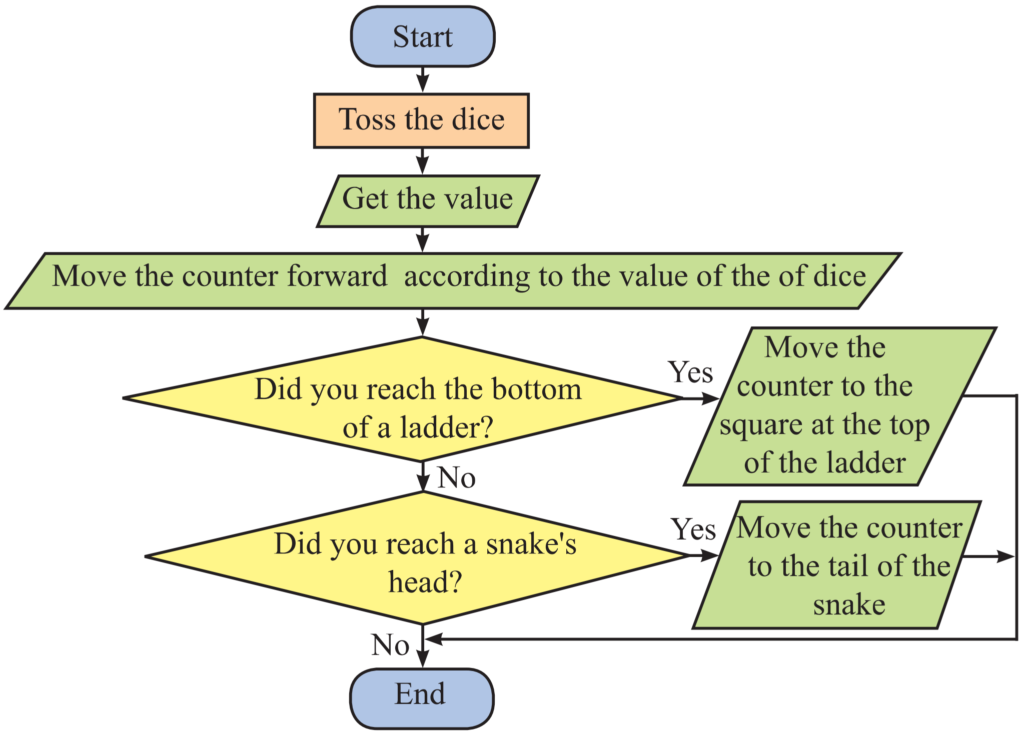
Each time the dice is tossed, the following instructions are to be followed;

1. Check the number shown on the top face of dice.
2. Shift the counter to face by the number shown on the top face of the dice.
3. If the counter reaches the bottom of a ladder, move it to reach the top of the ladder.
4. If the counter reaches the head of the snake, move down to where the tail is.



One has to follow these conditions and reach square 36 to become the winner.

Given below is the flowchart relevant to the above example.

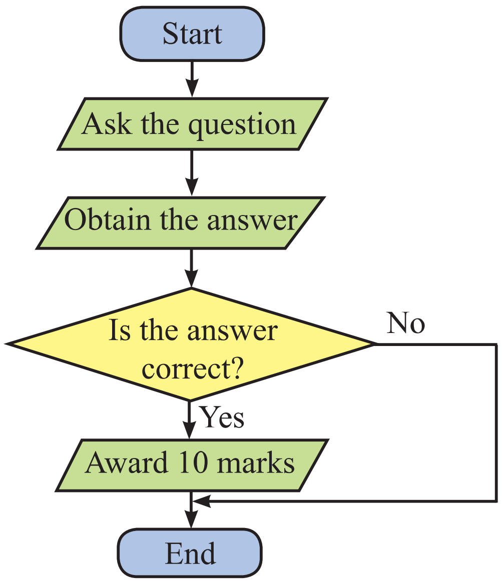


Start
Toss the dice
Get the value
Move the counter forward according to the value of the of dice
Did you reach the bottom of a ladder?
Yes
Move the counter to the square at the top of the ladder
No
Did you reach a snake's head?
Yes
Move the counter to the tail of the snake
No
End

Flowchart 3 : Snakes and Ladders game

Example 3 Consider a scenario where 10 marks are awarded to the correct answer.

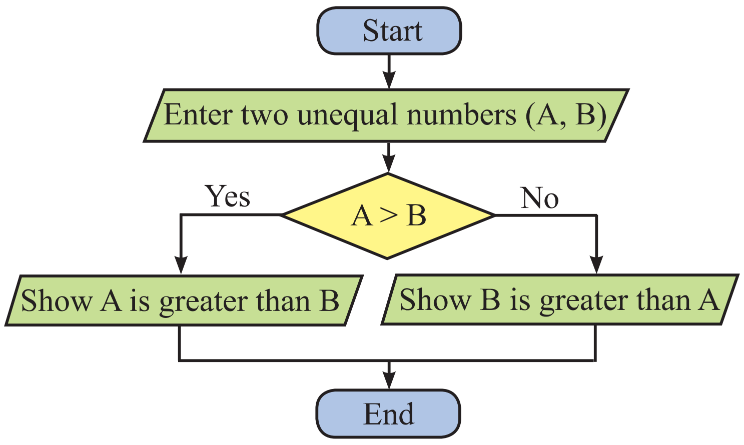
Before marks are awarded, it must be checked whether the answer is correct or wrong. If the answer is correct, 10 marks are awarded. Incorrect answers are awarded no marks. The above scenario can be represented by the following flowchart and control structures (Refer to Flowchart 4).



Flowchart 4 : Offering/Not offering marks for correct/incorrect answer

Example 4 Considering the scenario of finding the larger number from two unequal numbers.

Two numbers are given as input. Then the two numbers are compared. If the first is greater than the second, the output shows as the first number is greater. Otherwise, the output shows as the second number is greater (Refer to flowchart 5).



Flowchart 5 : Finding greater number

Refer to workbook for Activities 4.2 and 4.3

Note - You will learn about the repetition control structure in Grade 9.

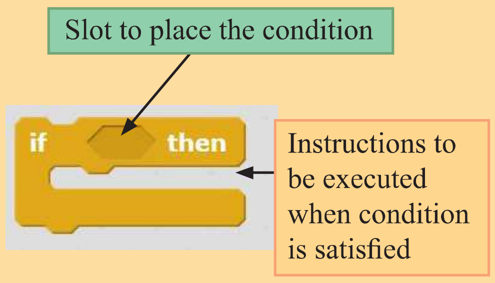
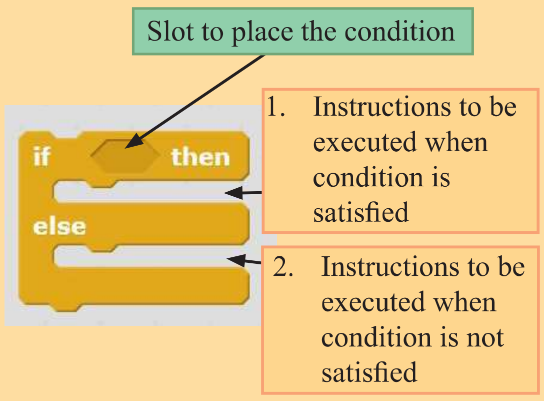
4.3 Selection control structures

Scratch is an Open Source Visual Programming Language produced to make programming easier to learn. Games, music, animations, interactive stories, etc. can be created using Scratch. Basic knowledge about Scratch was provided in the Grade 7 textbook.

Two types of selection control structures can be used in developing Scratch programs;

1. IF... THEN instructions block
2. IF... THEN... ELSE instructions block

Table 1 : Selection control structures

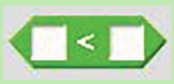
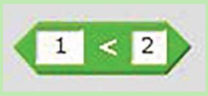
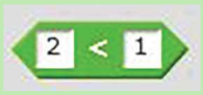
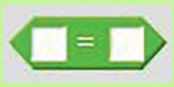
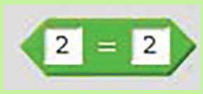
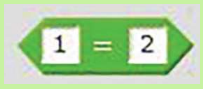
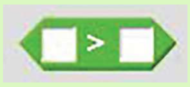
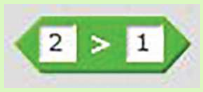
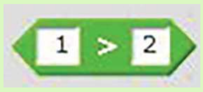
IF... THEN block	IF... THEN... ELSE... block
 Slot to place the condition if then Instructions to be executed when condition is satisfied	 Slot to place the condition if then else 1. Instructions to be executed when condition is satisfied 2. Instructions to be executed when condition is not satisfied
Instructions are executed only if the condition is satisfied.	The first instruction block is executed when the condition is satisfied. The second instruction block is executed when the condition is not satisfied.

Comparison blocks

There are instances where a decision has to be taken after comparing two values in programming. The decision is taken after comparing the two values; whether one value is greater/smaller/equal than other value.

Instruction blocks shown in the following table are used to compare values. These blocks output "True" or "False" after comparison.

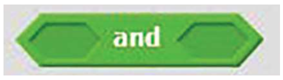
Table 2 : Comparison blocks

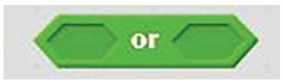
Instructions	Example	Output
 Check whether the value on the left is smaller than the value of the right	 1 < 2	True
	 2 < 1	False
 Check whether the value on the left equals to the right	 2 = 2	True
	 1 = 2	False
 Check whether the value on the left is greater than the one on the right	 2 > 1	True
	 1 > 2	False

Refer to workbook for Activity 4.4

Instruction with logical blocks

The following instruction blocks are used to combine comparison instruction blocks. There are three types of logical blocks as follows;

1. AND 

2. OR 

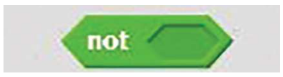
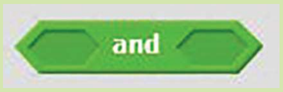
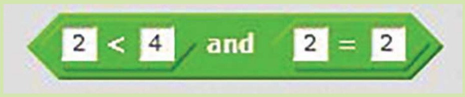
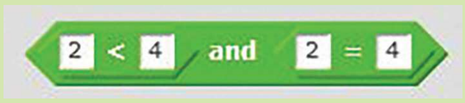
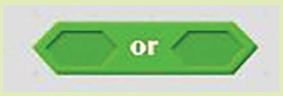
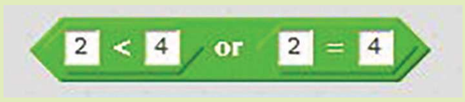
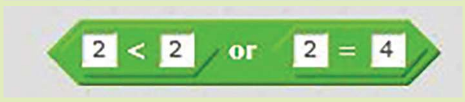
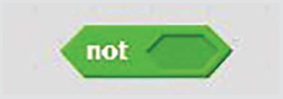
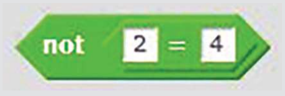
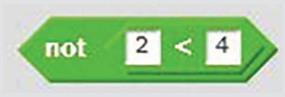
3. NOT 

Table 3 : Logical blocks

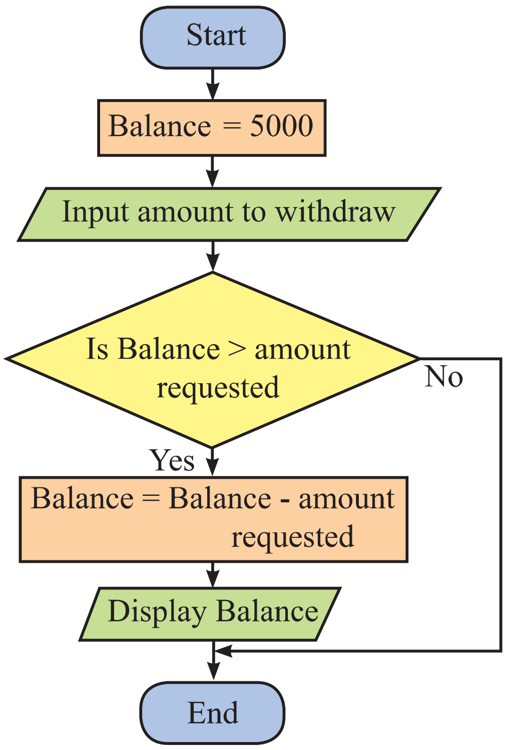
Instruction	Example	Reply
 If both expressions on left and right are true only, the output is true.	 	True False
 If both expressions on left and right are true or if only one is true, the output is true.	 	True False
 If the expression is false, the output is true. If the expression is true, the output is false.	 	True False

Refer to workbook for Activity 4.5

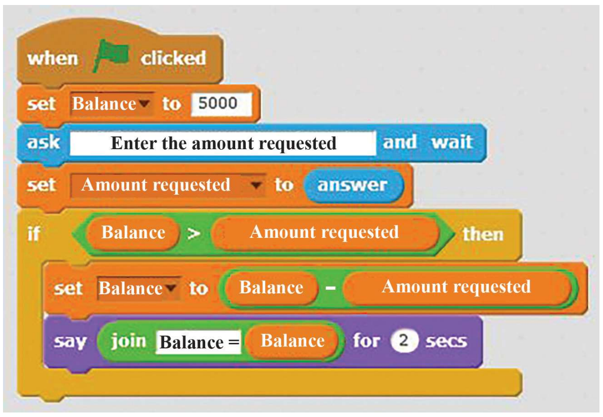
4.3.1 Developing Scratch program with selection control structure

Example 1 Display current balance after a withdrawal from an account with Rs.5000/=

In withdrawing money from an account, the current balance is first checked. Money is released only if the current balance is greater than the amount requested. The amount withdrawn is deducted from the current balance (Refer to Flowchart 6 and Scratch program 2).



Flowchart 6 : Display account balance

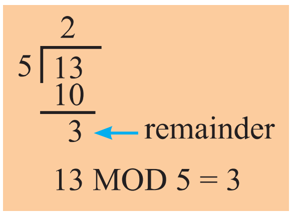


```
when [flag] clicked
set Balance to 5000
ask Enter the amount requested and wait
set Amount requested to answer
if Balance > Amount requested then
    set Balance to Balance - Amount requested
    say join Balance = Balance for 2 secs
```

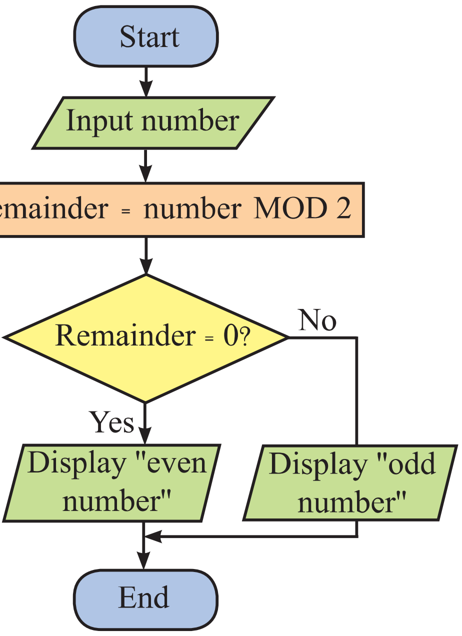
Scratch program 2 : Display account balance

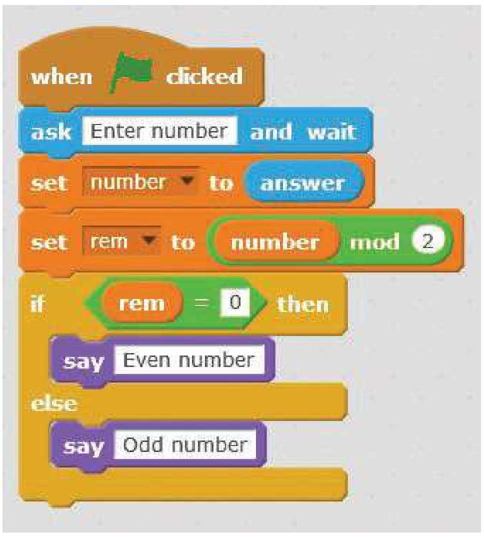
Example 2 Display whether an input number is odd or even.

Mathematical operator MOD is used to determine the remainder when a number is divided by another. For example, 13 MOD 5 is 3. When 13 is divided by 5, the remainder is 3.



Accordingly, if a number is divided by 2 and the remainder equals 0, it is an even number. If the remainder is 1, it is an odd number.



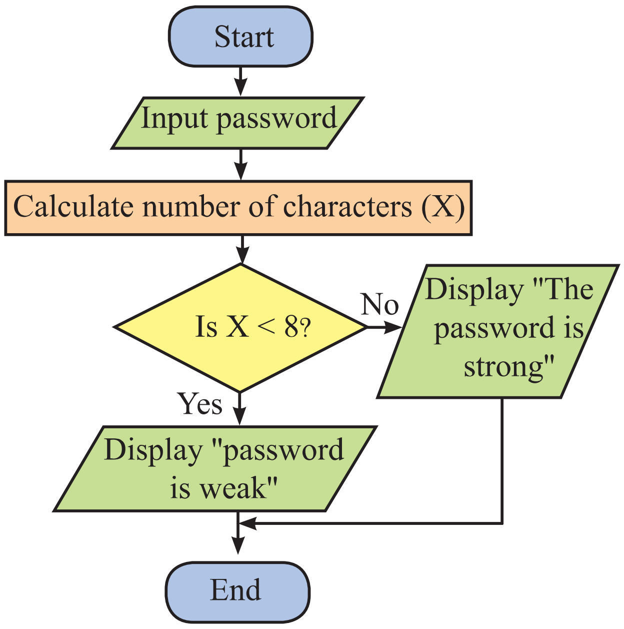


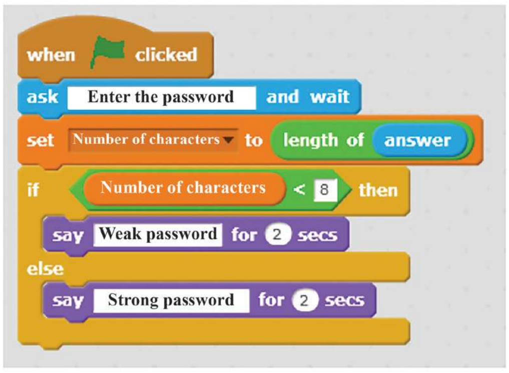
```
when 🏴 clicked
ask Enter number and wait
set number to answer
set rem to (number mod 2)
if rem = 0 then
    say Even number
else
    say Odd number
```

Scratch program 3 : Test whether a number is even /odd

Flowchart 7 : Text whether a number is even/ odd

Example 3 The number of characters in a password is one of indicators of strength. If the number of characters is less than 8, it is a weak password. If it is more than 8, it is a strong password (see Flowchart 8 and Scratch program 4).


Start
Input password
Calculate number of characters (X)
Is X < 8?
No
Display "The password is strong"
Yes
Display "password is weak"
End

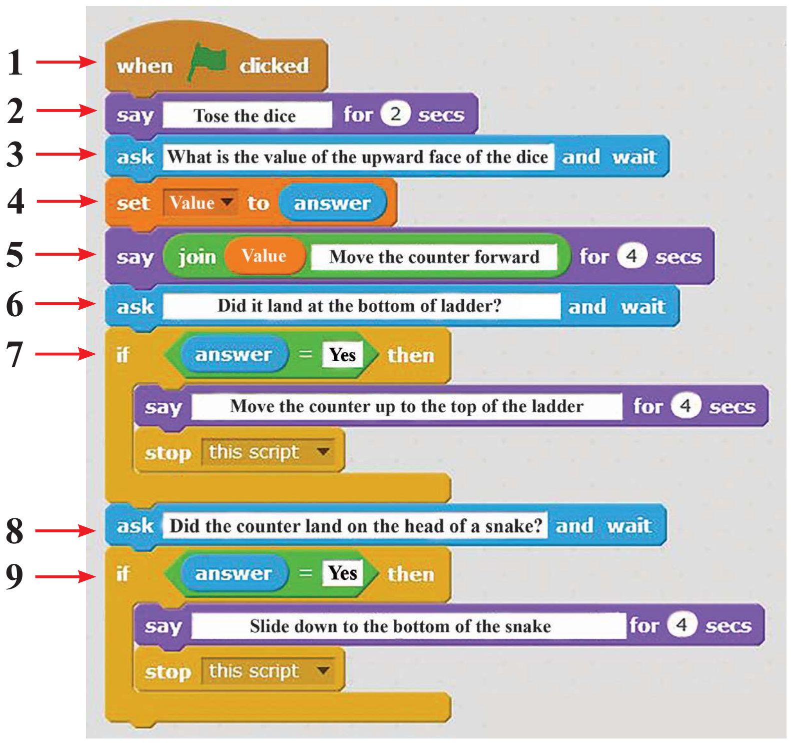


```
when 🏴 clicked
ask Enter the password and wait
set Number of characters to length of answer
if Number of characters < 8 then
    say Weak password for 2 secs
else
    say Strong password for 2 secs
```

Scratch programming 4 : Display whether a password is weak or strong

Flowchart 8 : Display whether a passwords is weak or strong

The Scratch program for *Snakes and Ladders* game is given below; (Refer to Scratch programs 5)



```
1 → when ⚑ clicked
2 → say [Tose the dice] for (2) secs
3 → ask [What is the value of the upward face of the dice] and wait
4 → set [Value ▾] to (answer)
5 → say (join (Value) [Move the counter forward]) for (4) secs
6 → ask [Did it land at the bottom of ladder?] and wait
7 → if <(answer) = [Yes]> then
        say [Move the counter up to the top of the ladder] for (4) secs
        stop [this script ▾]
8 → ask [Did the counter land on the head of a snake?] and wait
9 → if <(answer) = [Yes]> then
        say [Slide down to the bottom of the snake] for (4) secs
        stop [this script ▾]
```

Scratch program 5 : Snakes and Ladder game

Analysis of the program

1. Click on ⚑ to start.
2. First, display "Toss the dice".
3. Ask "What is the value?" and get the answer.
4. Assign the value obtained in step 03 above to the variable.
5. Declare 4 seconds to bring the counter forward according to the value of the dice.
6. Obtain "Yes" or "No" answer for the question, "Did it land at bottom of ladder?"
7. If the answer is "Yes", then move the counter to the square at the top of the ladder.
8. Display, "Did the counter come to the head of a snake?"
9. If the answer is "Yes", then move the counter to snake's tail.

Refer to workbook for Activities in 4.6, 4.7, 4.8 and 4.9

4.3.2 Applications for mobile and smart devices

Mobile and smart devices

Various applications are available for mobile and smart devices. These applications usually carry out tasks accurately and efficiently. Each mobile and smart device is developed for a specific function and they can be used according to the user requirements (See Figure 4.6).

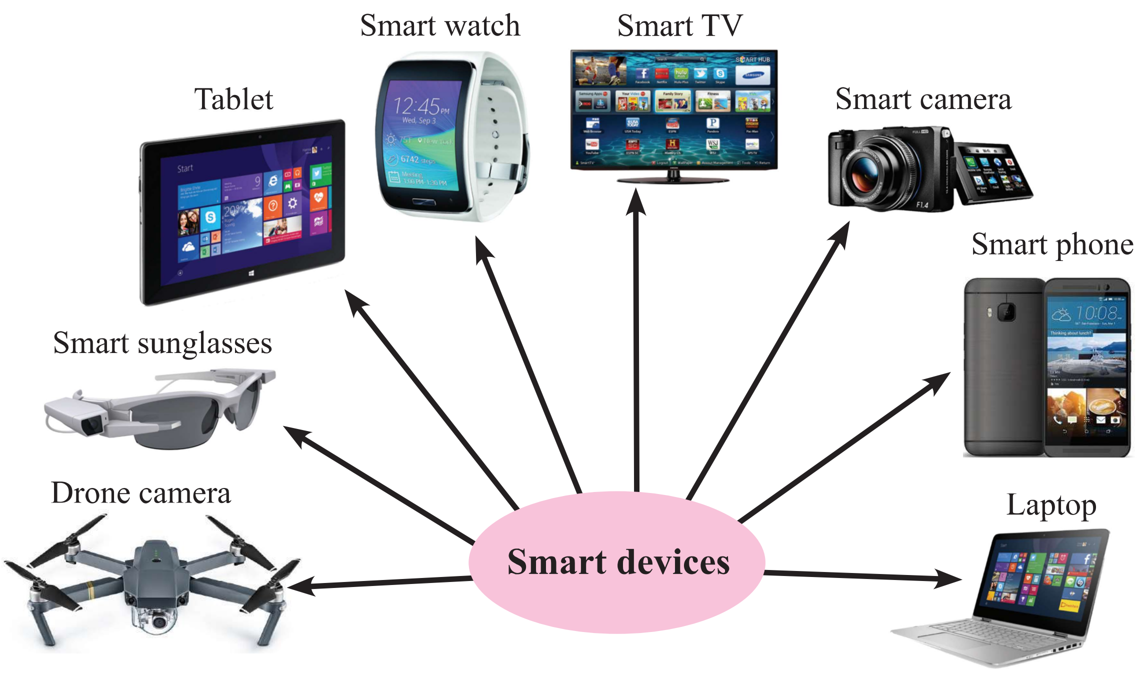


Figure 4.6 : Examples of mobile and smart devices

Tablets and laptops are widely used as general purpose machines. Other devices are meant for a specific functions. Therefore, it is necessary to determine whether the selected mobile device suits the intended purpose.

Table 4 : Smart devices and applications

Mobile device	Examples of use
Laptop computer	As a mobile device for general computer applications
Tablet computer	To surf the Internet and to take photos, etc.
Smart mobile phone	Telephone conversations, SMS and MMS messaging, taking photos, recording audio and video clips, surfing Internet, sending e-mails, etc.
Smart television	Managing and recording television programmes, e-mail and Internet
Smart camera	Taking photos, recording audio/video clips and sending them to other smart devices
Smart wristwatch	Displays time, sends SMS, sets alarm, accesses the Internet, etc.
Mobile spectacles	Watch scenery in 3D form, listen to audio, taking photos, record video clips, etc.
Drone camera	A remotely controlled, mini helicopter used to obtain aerial pictures or videos

Applications of mobile and smart devices

Many application software for mobile and smart devices are available on the Internet. Some application software can be downloaded free of charge while others have to be purchased. These downloaded software can be used after installing in the smart devices. The following shows some examples for application software for mobile and smart devices. (See Figure 4.7)

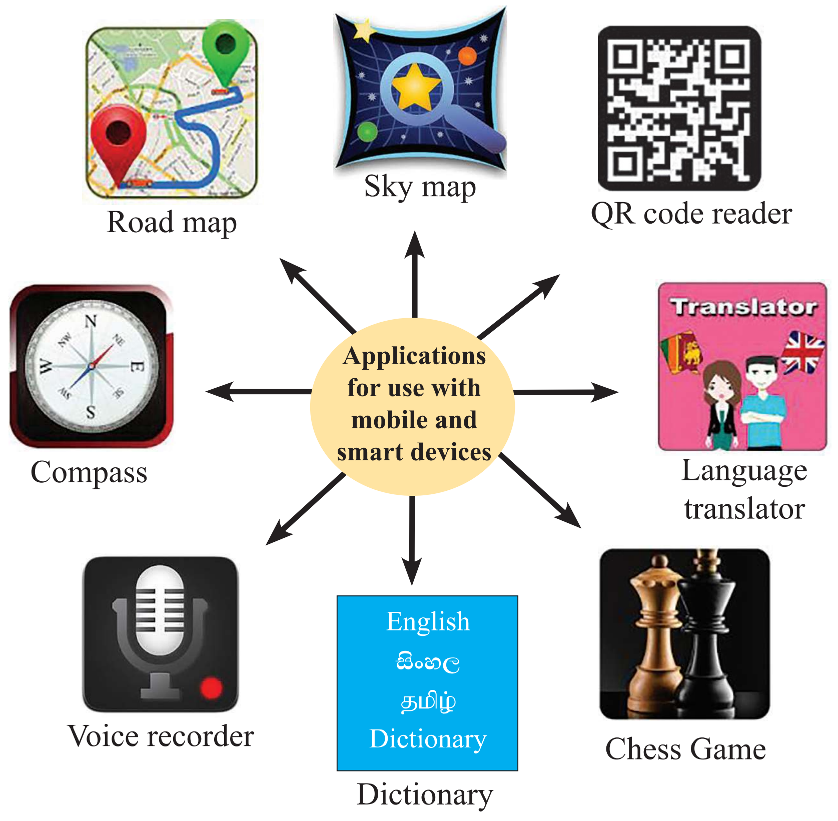


Figure 4.7 : Examples of the applications of mobile and smart devices

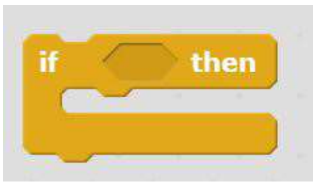
Table 5 : Mobile and smart devices and their uses

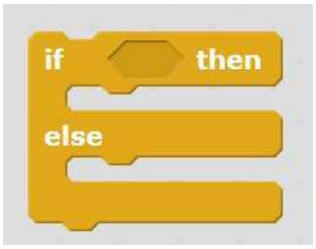
Smart device	Examples of use
Compass	finds the orientation
Route map	directs using GPS (finding routes), finds distance between two places, identifies traffic, etc.
Sky map	points the smart device towards a star or planet on the sky and see the details such as name, location, etc.
QR Code Reader	obtains information by scanning the QR code
Language translator	translates text in one language to another
Chess game	computer play as the opponent of Game
Sinhala dictionary	finds English term for Sinhala
Tape recorder	records and playback sound

Refer to workbook for the Activity 4.10

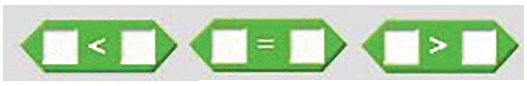
Summary

- Computer programs are developed to accept input, process data and produce output. Generally, the algorithm is written first and then the algorithm is converted to a program.
- There are three types of control structures that can be used in an algorithm;
  1. Sequence
  2. Selection
  3. Repetition
- A sequence follows steps in the algorithm the one after the other.
- In selection, the program selects the cause of action based on whether condition is satisfied or not. Scratch programming uses "if then" and "if then else" control structures for selection.

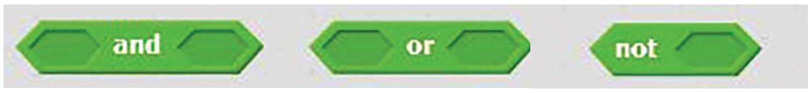




- Scratch uses three types of comparison blocks.



- Scratch uses three types of logical blocks.



- The "Repetition" control structures, will be taught in a future lesson.