2 ENCODING SCHEMES AND NUMBER SYSTEM (NCERT CLASS 11)

INTRODUCTION

- The mechanism of converting data into an equivalent cipher using specific code is called encoding.
- Cipher means something converted to a coded form to hide/conceal it from others. It is also called encryption (converted to cipher) and sent to the receiver who in turn can decrypt it to get back the actual content.

A computer can handle numeric and non numeric data like letters, punctuation marks and other special characters. Some pre defined codes are used to represent numeric and non numeric characters. There are various standard encoding schemes each part of data is assigned a unique code. Some of popular encoding schemes are mentioned below :

(I) **American Standard Code for Information Interchange (ASCII) :** Total number of different characters on theEnglish keyboard that can be encoded by 7-bit ASCII code is $2^7 = 128$.Out of 7 bits, 3 are zone bits and 4 are numeric bits. ASCII-8 can represent 256 characters. It is an extended form of ASCII-7.

(II) **Indian Script Code for Information Interchange (ISCII) :** A lot of efforts have gone into facilitating the use of Indian languages on computers. In 1991, the Bureau of Indian Standards adopted the ISCII. It is an 8-bit code representation for Indian languages which means it can represent 2^8=256 characters. It retains all 128 ASCII codes and uses rest of the codes (128) for additional Indian language character set. Additional codes have been assigned in the upper region (160– 255) for the 'aksharas' of the language.

(III) **UNICODE :** Unicode is a new universal coding standard adopted by all new platforms. It ispromoted by Unicode Consortium which is a non profit organization. Unicode provides a unique number for every character irrespective of the platform, program and the language. It is a character coding system designed to support the worldwide interchange, processing, and display of the written texts of the diverse languages. Commonly used UNICOD Eencodings areUTF-8, UTF-16 and UTF-32. It is a superset of ASCII, and the values 0–128 have the same character as in ASCII.

NUMBER SYSTEM

- Each number system has a base also called a Radix. A decimal number system is a system of base 10; binary is a system of base 2; octal is a system of base 8; and hexadecimal is a system of base 16. What are these varying bases? The answer lies in what happens when we count up to the maximum number that the numbering system allows. In base 10, we can count from 0 to 9, that is,10 digits.

Number System	Base	Symbols used
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F where A = 10; B = 11; C = 12; D = 13; E = 14; F = 15

(I) **Decimal Number System :** Decimal number system is known as base-10 system since 10 digits (0 to 9) are used.

(II) **Binary Number System :** Binary number system is formed by two digit 0 and 1.This system is also referred as base-2 system as it has two digits only. Some examples of binary numbers are 1001011, 1011.101, 111111.01. A binary number can be mapped to an equivalent decimal number that can be easily understood by the human.

Following table represent Binary value for (0–9) digits of decimal number system :

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

(III) **Octal Number System :** Octal number system was devised for compact representation of the binary numbers. Octal number system is called base-8 system as it has total eight digits (0-7), and positional value is expressed in powers of 8. Three

binary digits ($8=2^3$) are sufficient to represent any octal digit.

Following Table represent Decimal and binary equivalent of octal numbers 0 – 7:

Octal digit	Decimal value	3-bit Binary Number
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111

(IV) **Hexadecimal Number System :** Hexadecimal numbers are also used for compact representation of binary numbers. It consists of 16 unique symbols (0 – 9, A–F), and is called base-16 system. In hexadecimal system, each alphanumeric digit is represented as a group of 4 binary digits because 4 bits ($2^4=16$) are sufficient to represent 16 alphanumeric symbols.

Following table represent decimal and binary equivalent of hexadecimal numbers 0–9, A–F:-

HEXADECIMAL SYMBOL	DECIMAL VALUE	4-BIT BINARY NUMBER
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

- Applications of Hexadecimal Number System

(I) Usually, size of a memory address is 16-bit or 32-bit. To access 16-bit memory address, a programmer has to use 16 binary bits, which is difficult to deal with. To simplify the address representation, hexadecimal and octal numbers are used.

(II) Hexadecimal numbers are also used for describing the colours on the webpage. Each colour is made up of three primary colours red, green and blue, popularly called RGB (in short). In most colour maps, each colour is usually chosen from a palette of 16 million colours. Therefore, 24 bits are required for representing each colour having three components (8 bits for Red, 8 bits for Green, 8 bits for Blue component).

CONVERSION BETWEEN NUMBER SYSTEM

Converting a number from one Base to another:-

(I) **Binary to Decimal :** Method to convert Binary to Decimal:

1. Start at the rightmost bit.
2. Take that bit and multiply by 2n where n is the current position beginning at 0 and increasing by 1 each time. This represents a power of two.
3. Sum each terms of product until all bits have been used.

Example

Convert the Binary number 101011 to its Decimal equivalent.

$1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$

$32 + 0 + 8 + 0 + 2 + 1 = (43)_{10}$

Example

Convert the Binary number 1001 to its Decimal equivalent.

$1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0$

$8 + 0 + 0 + 1 = (9)_{10}$

Example

Convert $(11011.101)_2$ to decimal

$2^4\ 2^3\ 2^2\ 2^1\ .\ 2^0\ 2^{-1}\ 2^{-2}\ 2^{-3}$

1 1 0 1 1 1 0 1

$= (1\times2^4)+(1\times2^3)+(0\times2^2)+(1\times2^1)+(1\times2^0)+(1\times2^{-1})+(0\times2^{-2})+(1\times2^{-3})$

$= 16+8+0+2+1+0.5+0+0.125$

$= (27.625)_{10}$

(II) **Decimal to Binary :** Method to convert a Decimal number into its Binary equivalent:

1. Divide the decimal number by 2.
2. Take the remainder and record it on the side.
3. Divide the quotient by 2.
4. REPEAT UNTIL the decimal number cannot be divided further.
5. Record the remainders in reverse order and you get the resultant binary number.

Example

Convert the Decimal number 125 into its Binary equivalent.

125 / 2 = 62	1
62 / 2 = 31	0
31 / 2 = 15	1
15 / 2 = 7	1
7 / 2 = 3	1
3 / 2 = 1	1
1 / 2 = 0	1

Answer: $(1111101)_2$

Example

Convert $(105.15)_{10}$ to binary

Let us convert 105 first.

$(105)_{10} = (1101001)_2$

Let us convert $(0.15)_{10}$

Multiply 0.15 by 2	0.30
Multiply 0.30 by 2	0.60
Multiply 0.60 by 2	1.20
Multiply 0.20 by 2	0.40
Multiply 0.40 by 2	0.80
Multiply 0.80 by 2	1.60

Reading the integers from top to bottom $(0.15)_{10} = (0.001001)_2$

Final result $(105.15)_{10} = (1101001.001001)_2$

(III) **Decimal to Octal :** The method to convert a decimal number into its octal equivalent:

1. Divide the decimal number by 8.
2. Take the remainder and record it on the side.
3. Divide the quotient by 8.
4. REPEAT UNTIL the decimal number cannot be divided further.
5. Record the remainders in reverse order and you get the resultant binary

Example

Convert the Decimal number 125 into its Octal equivalent.

125 / 8 = 15	5
15/ 8 = 1	7
1/8 =0	1

Answer: $(175)_8$

Example

Convert $(0.75)_{10}$ to Octal

Multiply the given fraction by 8. Keep the integer in the product as it is and multiplythe new fraction in the product by 8. Continue the process and read the integers in the products from top to bottom.

Given fraction: 0.75

Multiply 0.75 by 8: 6.00

Reading the integers from top to bottom 0.75 in decimal number system is 0.6 in octal number system.

(IV) **Octal to Decimal :** Method to convert Octal to Decimal:

1. Start at the rightmost bit.
2. Take that bit and multiply by 8n where n is the current position beginning at 0 and increasing by 1 each time. This represents the power of 8.
3. Sum each of the product terms until all bits have been used.

Example

Convert the Octal number 321 to its Decimal equivalent.

$3 * 8^2 + 2 * 8^1 + 1 * 8^0$

$192+16+1 = (209)_{10}$

Example

Convert $(23.25)_8$ to decimal

$8^1 \quad 8^0 . \quad 8^{-1} \quad 8^{-2}$

$2 \quad 3 \quad 2 \quad 5$

$= (2 \times 8^1)+ (3 \times 8^0)+ (2 \times 8^{-1})+ (5 \times 8^{-2})$

$= 16+3+0.25+0.07812$

$= (19.32812)_{10}$

(V) **Decimal to Hexadecimal :** Method to convert a Decimal number into its Hexadecimal equivalent:

1. Divide the decimal number by 16.
2. Take the remainder and record it on the side.
3. REPEAT UNTIL the decimal number cannot be divided further.
4. Record the remainders in reverse order and you get the equivalent hexadecimal number.

Example

Convert the Decimal number 300 into its hexadecimal equivalent.

300 / 16 = 18	12-(C)
18 / 16 = 1	2
1 / 16 = 0	1

Answer: $(12C)_{16}$

Example

Convert $(0.75)_{10}$ to hexadecimal

Multiply the given fraction by 16. Keep the integer in the product as it is and multiply the new fraction in the product by 16. Continue the process and read the integers in the products from top to bottom.

Given fraction 0.75

Multiply 0.75 by 16 12.00 - C

Reading the integers from top to bottom 0.75 in decimal number system is 0C in Hexadecimal number system.

(VI) **Hexadecimal to Decimal :** Method to convert Hexadecimal to Decimal:

1. Start at the rightmost bit.
2. Take that bit and multiply by 16n where n is the current position beginning at 0 and increasing by 1 each time. This represents a power of 16.
3. Sum each terms of product until all bits have been used.

Example

Convert the Hexadecimal number AB to its Decimal equivalent.

$=A * 16^1 + B * 16^0$

$=10 * 16^1 + 11 * 16^0$

$=160+11 = (171)_{16}$

Example

Convert $(1E.8C)_{16}$ to decimal

$16^1\ 16^0\ .\ 16^{-1}\ 16^{-2}$

1E 8 C

$= (1 \times 16^1)+ (14 \times 16^0)+ (8 \times 16^{-1})+ (12 \times 16^{-2})$

$= 16+14+0.5+0.04688$

$= (30.54688)_{10}$

(VII) **Binary to Hexadecimal :** The hexadecimal number system uses the digits 0 to 9 and A, B, C, D, E, F. Method to convert a Binary number to its Hexadecimal equivalent is:

We take a binary number in groups of 4 and use the appropriate hexadecimal digit init's place. We begin at the rightmost 4 bits. If we are not able to form a group of four,insert 0s to the left until we get all groups of 4 bits each. Write the hexadecimal equivalent of each group. Repeat the steps until all groups have been converted.

Example

Convert the binary number 1000101 to its Hexadecimal equivalent.

0100 0101

4 5

Note that we needed to insert a 0 to the left of 100.

Answer: $(45)_{16}$

In case of a fractional binary number form groups of four bits on each side of decimal point. Then replace each group by its corresponding hexadecimal number.

Example

Convert $(11100.1010)_2$ to hexadecimal equivalent.

0001 1100 . 1010

1 C . A

Answer : $(1C.A)_{16}$

(VIII) **Hexadecimal to Binary :**Method to convert a Hexadecimal number to its Binary equivalent is: Convert each digit of Hexadecimal Number to it's binary equivalent and write them in 4 bits. Then, combine each 4 bit binary number and that is the resulting answer.

Example

Convert the Hexadecimal number $(10AF)_{16}$ to its Binary equivalent.

1 0 A F

0001 | 0000 | 1010 | 1111

Answer: $(0001000010101111)_2$

Example

Convert the Hexadecimal number $(A2F)_{16}$ to its Binary equivalent.

A 2 F

1010 | 0010 | 1111

Answer: $(1010\ 0010\ 1111)_2$

(IX) **Binary to Octal and Octal to Binary :** To convert Binary to Octal, as the octal system is a power of two (2^3), we can take the bits into groups of 3 and represent each group as an octal digit. The steps are the same for the binary to hexadecimal conversions except we are dealing with the octal base now. To convert from octal to binary, we simply represent each octal digit in it's three bit binary form.

Example

Convert the Octal number $(742)_8$ to its Binary equivalent.

7 | 4 | 2

111 | 100 | 010

Answer: $(111100010)_2$

(X) **Hexadecimal to Octal and Octal to Hexadecimal :** To convert Hexadecimal to Octal, Convert each digit of Hexadecimal Number to it's binary equivalent and write them in 4 bits. Then, combine each 3 bit binary number and that is converted into octal.

Example

Convert the Hexadecimal number $(A42)_{16}$ to its Octal equivalent.

A | 4 | 2

1010 | 0100 | 0010

101 | 001 | 000 | 010

Answer: $(5102)_8$

To convert Octal to hexadecimal, convert each digit of Octal Number to it's binary equivalent and write them in 3 bits. Then, combine each 4 bit binary number and that is converted into hexadecimal.

Example

Convert the Octal number $(762)_8$ to its hexadecimal equivalent.

7 | 6 | 2
101 | 110 | 010
0001 | 0111 | 0010

Answer: $(172)_{16}$

BINARY REPRESENTATION OF INTEGERS

Binary number can be represented only by using 0's and 1's, but can not use the sign (–) to denote the negative number or sign (+) to denote the positive number. So it must be either 0 or 1. There are three methods to represent binary number. They are:-

(I) **Sign and magnitude method :** In this method, first bit is considered as a sign bit. Here positive number starts with 0 and negative number starts with 1.

Example:- Take a number 25.

25/2 = 12	1
12/2 = 6	0
6/2 = 3	0
3/2 = 1	1
1/2 = 0	1

So the binary number is $(11001)_2$. If we take the size of the word is 1 byte, then the number 25 will be represented as 00011001.

Suppose, if the number is -25, and then it will be represented as 10011001.

(II) **One's complement method :** In this method, the positive number is represented as same as the binary number. If the number is negative, then we need to find one's complement of a binary number. The one's complement of a binary number will replace every 0 with 1 and vice- versa.

Example

(a) Represent 86 in one's complement method (I byte representation)

86/2 =43	0
43/2 =21	1
21/2 =10	1
10/2 =5	0
5/2 =2	1
2/2 =1	0
1/2 =0	1

The binary number is 1010110.

1 byte representation of number 86 is 01010110.

(b) Represent -55 in one's complement method (I byte representation)

55/2 =27	1
27/2 =13	1
13/2 =6	1
6/2 =3	0
3/2 =1	1
1/2 =0	1

The binary number is 110111.

1 byte representation is 00110111

The given number is negative; hence we need to calculate one's complement

One's complement of 00110111 is 11001000 (convert 1 into 0 and 0 into 1)

Thus, the 1 byte representation of number -55 is 11001000.

(III) **Two's complement method:** In this method, the positive number is represented as the binary number. If the number is negative, then we need to calculate two's complement of a binary number. The two's complement of a binary number is calculated by adding 1 to its one's complement.

Example

(a) Represent 87 in two's complement method (I byte representation)

87/2 =43	1
43/2 =21	1
21/2 =10	1
10/2 =5	0
5/2 =2	1
2/2 =1	0
1/2 =0	1

The binary number is 1010111

Hence, the 1 byte representation of number 86 is 01010111.

(b) Represent -54 two's complement method (I byte representation)

54/2 =27	0
27/2 =13	1
13/2 =6	1
6/2 =3	0
3/2 =1	1
1/2 =0	1

The binary number is 110110

Hence, the 1 byte representation is 00110110

The given number is negative; hence we need to calculate two's complement.

One's complement of 00110110 is 11001001 (convert 1 into 0 and 0 into 1).

Add 1 to one's complement

```
      1
11001001 (1+1=2, binary equivalent =11)
    + 1
-----------
11001010
-----------
```

Thus, 1 byte representation of number -54 is 11001010 □□□

4 INTRODUCTION TO PROBLEM SOLVING (NCERT CLASS 11)

INTRODUCTION

The success of a computer in solving a problem depends on how correctly and precisely we define the problem, design a solution (algorithm) and implement the solution (program) using a programming language. Thus, problem solving is the process of identifying a problem, developing an algorithm for the identified problem and finally implementing the algorithm to develop a computer program.

STEPS FOR PROBLEM SOLVING

When problems are straight forward and easy, we can easily find the solution. But a complex problem requires a methodical approach to find the right solution. In other words, we have to apply problem solving techniques. Problem solving begins with the precise identification of the problem and ends with a complete working solution in terms of a program or software. Key steps required for solving a problem using a computer are discussed below:-

(I) **Analysing the problem :** We need to read and analyse the problem statement carefully in order to list the principal components ofthe problem and decide the core functionalities that our solution should have. By analysing a problem, we would be able to figure out what are the inputs tha tour program should accept and the outputs that itshould produce.

(II) **Developing an Algorithm :** Before writing a program code for a given problem, it is essential to device a solution. The solution is represented in natural language and is called an algorithm.

(III) **Coding :** After finalising the algorithm, we need to convert the algorithm into the format which can be understood by the computer to generate the desired solution. Different high level programming languages can be used for writing a program.

(IV) **Testing and Debugging :** The program created should be tested on various parameters. The program should meet the requirements of the user. It must respond within the expected time. It should generate correct output for all possible inputs. In the presence of syntactical errors, no output will be obtained. In case the output generated is incorrect, then the program should be checked for logical errors, if any.

Software programs goes through testing, updating, troubleshooting, and maintenance during the development process. Usually, software contains errors and bugs, which are removed routinely. Debugging is the process of fixing a bug in the software.

ALGORITHM

An algorithm is an effective method expressed as a finite list of well definedinstructions for calculating a function, starting from an initial state and initial input. The instructions describe a computation, which will eventually produce output, when executed. We can use algorithm to solve any kind of problems. However, before writing a program, we need to write the steps to solve the problem in simple English language. This step-by-step procedure to solve the problem is called algorithm.

The purpose of using an algorithm is to increase the reliability, accuracy and efficiency of obtaining solutions.

- **Characteristics of a good algorithm :**
  - (I) Precision — the steps are precisely stated or defined.
  - (II) Uniqueness — results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
  - (III) Finiteness — the algorithm always stops after a finite number of steps.
  - (IV) Input — the algorithm receives some input.
  - (V) Output — the algorithm produces some output.
- While writing an algorithm, it is required to clearly identify the following:
  - (I) The input to be taken from the user.
  - (II) Processing or computation to be performed to get the desired result.
  - (III) The output desired by the user.

REPRESENTATION OF ALGORITHMS

The software designers or programmers analyse the problem and identify the logical steps that need to be followed to reach a solution by using theiralgorithmic thinking skills. There are two common methods of representing an algorithm —flowchart and pseudocode. This methods can be used to represent an algorithm while keeping in mind the following:

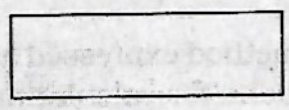
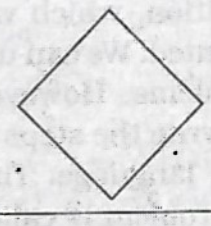
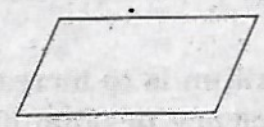
(I) It showcases the logic of the problem solution, excluding any implementational details.

(II) It clearly reveals the flow of control during execution of the program.

- **Flowchart — Visual Representation of Algorithms :**

A flowchart is a visual representation of an algorithm. A flowchart is a diagram made up of boxes, diamonds and other shapes, connected by arrows. Each shape represents a step of the solution process and the arrow represents the order or link among the steps.

Table : Shapes or symbols to draw flow charts :

Flowchart Symbol	Function	Description
	Start/Ends	Also called "Terminator" symbol. It indicates where the flow starts and ends.
	Process	Also called "Action Symbol," it represents a process action, or a single step.
	Decision	A decision or branching point, usually a yes/no or true/false question is asked, and based on the answer, the path gets split into two branches.
	Input/Output	Also called data symbol, this parallelogram shape is used to input or output data
	Arrow	Connector to show order of flow between shapes.

The way of execution of the program shall be categorized into three ways:

(I) sequence statements; (II) selection statements; and (IIII) iteration or looping statements. This is also called as "control structure".

(I) Sequence statements: In this program, all the instructions are executed one after another.

Example

Write an algorithm to print "Good Morning".

Step 1: Start

Step 2: Print "Good Morning"

Step 3: Stop.

Example

Write an algorithm to find area of a rectangle.

Step 1: Start

Step 2: Take length and breadth and store them as L and B?

Step 3: Multiply by L and B and store it in area

Step 4: Print area

Step 5: Stop

In the above mentioned two examples, all the instructions areexecuted one after another. These examples are executed under sequential statement.

(II) **selection statements :** In this program, some portion of the program is executed based upon the conditional test. If the conditional test is true, compiler will execute some part of the program, otherwise it will execute the other part of the program.

Example

Write an algorithm to check whether he is eligible to vote? (more than or equal to 18 years old).

Step 1: Start

Step 2: Take age and store it in age

Step 3: Check age value, if age >= 18 then go to step 4 else step 5

Step 4: Print "Eligible to vote" and go to step 6

Step 5: Print "Not eligible to vote"

Step 6: Stop

Example

Write an algorithm to check whether given number is +ve, -ve or zero.

Step 1: Start

Step 2: Take any number and store it in n.

Step 3: Check n value, if n > 0 then go to step 5 else go to step 4

Step 4: Check n value, if n < 0 then go to step 6 else go to step 7

Step 5: Print "Given number is +ve" and go to step 8

Step 6: Print "Given number is -ve" and go to step 8

Step 7: Print "Given number is zero"

Step 8: Stop

In the above mentioned examples, all the statements are not executed, but

based upon the input, some portions of the algorithm are executed, because we have "true" or "false" situation in the program.

(III) **Iterative statements :** In some programs, certain set of statements are executed again and again based upon conditional test. i.e. executed more than one time. This type of execution is called "looping or iteration".

Example

Write an algorithm to print all natural numbers up to „n?.

Step 1: Start

Step 2: Take any number and store it in n.

Step 3: Store 1 in I

Step 4: Check I value, if I<=n then go to step 5 else go to step 8

Step 5: Print I

Step 6: Increment I value by 1

Step7: Go to step 4

Step 8: Stop

In the above example, steps 4, 5, 6 and 7 are executed more than one time.

- **Pseudocode :** A pseudocode (pronounced Soo-doh-kohd) is another way of representing an algorithm. It is considered as a non-formal language that helps programmers to write algorithm. It is a detailed description of instructions that a computer must follow in a particular order. It is intended for human reading and cannot be executed directly by the computer. No specific standard for writing a pseudocode exists. The word "pseudo" means "not real," so "pseudocode" means "not real code". Following are some of the frequently used keywords while writing pseudocode:

(I)	INPUT	(II)	COMPUTE
(III)	PRINT	(IV)	INCREMENT
(V)	DECREMENT	(VI)	IF/ELSE
(VII)	WHILE	(VIII)	TRUE/FALSE

There is no strict set of standard notations for pseudocode, but some of the most widely recognised are:

(I) INPUT/READ – indicates a user will be inputting something.

(II) OUTPUT/WRITE – indicates that an output will appear on the screen.

(III) WHILE – a loop (iteration that has a condition at the beginning).

(IV) FOR – a counting loop (iteration).

(V) REPEAT – UNTIL – a loop (iteration) that has a condition at the end.

(VI) IF – THEN – ELSE – a decision (selection) in which a choice is made any instructions that occur inside a selection or iteration are usually indented.

Representation of algorithm using Pseudo-code:

Example-

Pseudocode to find out largest of two numbers.

```
Write "please enter 2 numbers"
Read n1,n2
If(n1 > n2)
Set max to n1
Else
Set max to n2
```

Benefits of Pseudocode :

(I) A pseudocode of a program helps in representing the basic functionality of the intended program.

(II) By writing the code first in a human readable language, the programmer safeguards against leaving out any important step.

(III) Besides, for non-programmers, actual programs are difficult to read and understand, but pseudocode helps them to review the steps to confirm that the proposed implementation is going to achieve the desire output.

FLOW OF CONTROL

The flow of control depicts the flow of events as represented in the flow chart. The events can flow in a sequence, or on branch based on a decision or even repeat some part for a finite number of times.

(I) **SEQUENCE :** Such algorithms where all the steps are executed oneafter the other are said to execute in sequence. However, statements in an algorithm may not always execute in a sequence. We may sometimes require the algorithm to either do some routine tasks in a repeated manner or behave differently depending on the outcomes of previous steps.

(II) **SELECTION :** Selecting which path of an algorithm to execute depending on some criteria. For example, if you passed a class in school, then we execute the operations that clap and cheer and play a song. But if you didn't pass the class, then maybe we would say, "Better luck next time, hang in there!"

The program checks one or more conditions and perform operations (sequence of actions) depending on true or false value of the condition. These true or false values are called binary values.

Conditionals are written in the algorithm as follows:

If <condition> then

steps to be taken when the

condition is true/fulfilled

There are situations where we also need totake action when the condition is not fulfilled. To represent that, we can write:

If <condition> is true then

steps to be taken when the condition is true/ fulfilled

otherwise steps to be taken when the condition is false/not fulfilled

In programming languages, 'otherwise' is represented using Else keyword. Hence, a true/false conditional is written using if-else block in actual programs.

(III) **REPETITION :** The kind of statements we use, when we want something to be done repeatedly, for a given number of times. Forexample , suppose 10 cards need to be withdrawn in the card game , then the pseudocode needs to be repeated 10 times to decide the winner.

In programming, repetition is also known as iteration or loop. A loop in an algorithm means execution of some program statements repeatedly till some specified condition is satisfied. Many times, we want to be able to repeat a set of operations a specific number of times or until some condition occurs.

VERIFYING ALGORITHMS

Today software are used in more critical services — like in the medical field or in space shuttles. Such software needs to work correctly in every situation. Therefore, the software designer should make sure that the functioning of all the components are defined correctly, checked and verified in every possible way.

When we have written an algorithm,we want to verify that it is working as expected. To verify, we have to take different input values and go through all the steps of the algorithm to yield the desired output for each input value and may modify or improve as per the need. The method of taking an input and running through the steps of the algorithm is sometimes called dry run. Such a dry run will help us to:

(I) Identify any incorrect steps in the algorithm.

(II) Figure out missing details or specifics in the algorithm.

Suppose we develop some software without verifyingthe underlying algorithm and if there are errors in the algorithm, then the software developed will not run. Hence, it is important to verify an algorithm since the effort required to catch and fix a mistake is minimal.

COMPARISON OF ALGORITHM

There can be more than one approach to solve a problem using computer and hence we can have more than one algorithm.

Algorithms can be compared and analysed on the basis of the amount of processing time they need to run and the amount of memory that is needed to execute the algorithm. These are termed as time complexity and space complexity, respectively. The choice of an algorithm over another is done depending on how efficient they are in terms of proecssing time required (time complexity) and the memory they utilise (space complexity).

CODING

Once an algorithm is finalised, it should be coded in a high-level programming language as selected by the programmer. The ordered set of instructions are written in that programming language by following its syntax. Syntax is the set of rules or grammar that governs the formulation of the statements in the language, such as spellings, order of words, punctuation, etc.

Coding is basically implementing logic/algorithm/ pseudocode (derived from requirement analysis/problem definition) in one of the preferred programming language(C,C++ Java, Javascript, python etc) as per the protocols/rules/syntactic grammar of the choosen language by following the design decisions.

A program written in a high-level language is called source code. We need to translate the source code into machine language using a compiler or an interpreter, so that it can be understood by the computer.

DECOMPOSITION

Sometimes a problem may be complex, that is, its solution is not directly derivable. In such cases, we need to decompose it into simpler parts.

The basic idea of solving a complex problem by decomposition is to 'decompose' or break down a complex problem into smaller sub problems. These sub problems are relatively easier to solve than the original problem. Finally, the subproblems are combined in a logical way to obtain the solution for the bigger, main problem.

Once the individual sub problems are solved, it is necessary to test them for their correctness and integrate them to get the complete solution.

□□□