**Chapter 1**

**PROBLEM SOLVING**

***LEARNING OBJECTIVES***

*After going through this chapter, the readers will be able to:*

*apply problem solving techniques;* 

*define an algorithm and its features;*

*describe the analysis of the algorithm efficiency;* 

*discuss the analysis of algorithm complexity; and*

*design flowchart*

**1.1 INTRODUCTION**

In our daily life, we routinely encounter and solve problems. We pose problems that we need or want to solve. For this, we make use of available resources, and solve them. Some categories of resources include: the time and efforts of yours and others; tools; information; and money. Some of the problems that you encounter and solve are quite simple. But some others may be very complex.

In this unit we introduce you to the concepts of problem-solving, especially as they pertain to computer programming.

The problem-solving is a skill and there are no universal approaches one can take to solving problems. Basically one must explore possible avenues to a solution one by one until she/he comes across a right path to a solution. In general, as one gains experience in solving problems, one develop one’s own techniques and strategies, though they are often intangible. Problem-solving skills are recognized as an integral component of computer programming. It is a demand and intricate process which is equally important throughout the project life cycle especially – study, designing, development, testing and implementation stages. The computer problem solving process requires:

i) Problem anticipation

ii) Careful planning

iii) Proper thought process

iv) Logical precision

v) Problem analysis

vi) Persistence and attention.

At the same time it requires personal creativity, analytic ability and expressions. The chances of success are amplified when the problem solving is approached in a systematic way and satisfaction is achieved once the problem is satisfactorily solved. The problem should be anticipated as far as possible and properly defined to help the algorithm defined and development process.

Computer is a very powerful tool for solving problems. It is a symbol-manipulating machine that follows a set of stored instructions called a program. It performs these manipulations very quickly and stores the input, all commands and output. A computer can’t think like human so that when solving any problem we have to specify the needed initial data, the operations which to be performed. And the result you wanted as output. Any of the instructions missing you will get either no result or invalid result. In either case your problem has not yet been solved. Therefore e several step need to be considered before writing a program. These steps may free you from hours of finding and removing errors in your program (a process called debugging).It should also make the act of problem solving with a computer a much simpler task.

All types of computer programs are collectively referred to as software. Programming languages are also part of it. Physical computer equipment such as electronic circuitry, input output devices, storage media etc. comes under hard ware. Software governs the functioning of hardware. Operations performed by software may be built in to the hardware, while instructions executed by the hardware may be generated in software .

***~~Problem solving~~* ~~1~~**

The decision to incorporate certain functions in the hardware and others in the software is made by the manufacturer and designer of the software and hardware. Normal considerations for this are: cost, speed, memory required, adoptability and reliability of the system. Set of instructions of the high-level language used to code a problem to find its solution is referred to as source program. A translator program called a compiler or interpreted, translate the source program in to the object program. This is the compilation or interpretation phase. All the testing of the source programs as regards the correct format of instructions is performed at this stage and the errors, is any, and is printed. If there is no error the source program is transformed in to the machine language program called object program. The object program is executed to perform calculations this stage is the execution phase. Data, if required by the program are supplied now and the result is obtained on the output device.

**Source program**

**Computer System**

**Object Program**

**Results**

**Data if required**

**1.2 PROBLEM SOVING THECHNIQUES**

Problem solving is a process which defines systematization and mechanization. There are a number of steps that can be taken to raise the level of one’s performance in problem solving.

Steps for problem solving

A problem solving technique follows certain steps in finding the solution to a problem. Let us look in to the steps one by one.

**1.2.1 Problem definition phase**

The success in solving any problem is possible only after the problem has been fully understood. That is we cannot hope to solve problem, which we do not understand .So, the problem understanding is the first step towards the solution of the problem. In the problem definition phase, we must emphasize what must be done rather than how is is to be done. That is we try to extract precisely defined set of tasks from the problem statement. In experienced problem solvers too often gallop a head with the task of problem solving only to find that they are either solving the wrong problem or solving just one particular problem.

**1.2.2 Getting started on a problem**

There are many ways of solving a problem and there may be several solutions. So it is difficult t to recognize immediately which path could be more protective. Sometimes you do not have any idea where to begin solving a problem, even if the problem has been defined .Such block sometimes occurs because you are overly concerned with the details of the implementation even before you have completely understood or worked out a solution. The best advice is not to get concerned with the details. Those can come later when the intricacies of the problem has been understood.

**1.2.3 The use of specific examples**.

To get started on a problem, we can make use of heuristics that is the rule of thumb. This approach will allow us to start on the problem by picking a specific problem we wish to solve and try to work out the mechanism that will allow solving this particular problem. It is usually much easier to work out the details of a solution to a specific problem because the relationship between the mechanism and the

***Problem solving* 2**

problem is more clearly defines. This approach of focusing on a particular problem can give us the foothold we need for making a star on the solution to the general problem.

**1.2.4 Similarities among problems**

One way to make a start is by considering a specific example .Another approach is to bring the experience to bear on the current problem. So it is important to see if there are any similarities between the current problem and the past problems which we have solved. He more experience one has the more tools and techniques one can bring to bear in tackling the given problem. But some time it blocks us from discovering desirable or better solution to the problem. A skill that is important to try to develop in problem solving is the ability to view a problem from a variety of angles. One must be able to metaphorically turn a problem upside down, Inside out, sideways, backwards, forwards and so on. Once one has developed this skill it should be possible to get started on any problem.

**1.2.5 Working backwards from the solution**

In some cases we can assume that we already have the solution to the problem and then try to work backwards to the starting point. Even a guess at the solution to the problem may be enough to give us a foothold to start on the problem. We can systematize the investigation and avoid duplicate efforts by writing down the various steps taken and explorations made. Another practice that helps to develop the problem solving skills is , once we has solved problem , to consciously reflect back on the way we went about discovering the solution.

**1.3 USING COMPUTER AS A PROBLEM-SOLVING TOOL**

The computer is a resource-a versatile tool-that can help you solve some of the problems that you encounter. A computer is a very powerful general-purpose tool. Computers can solve or help to solve many types of problems. There are also many ways in which a computer can enhance the effectiveness of the time and effort that you are willing to devote to solving a problem. Thus it will prove to be well worth the time and effort you spend to learn how to make effective use of this tool.

In this section, we discuss the steps involved in developing a program. Program development is a multi-step process that requires you to understand the problem, develop a solution, write the program, and then test is. This critical process determines the overall quality and success of your program. If you carefully design each program using good structured development techniques our programs will be efficient, error free, and easy to maintain. The following are the steps in detail:

i) Develop an Algorithm and a Flowchart

ii) Write the program in a computer language (for example say C programming language) iii) Enter the program sing some editor.

iv) Test and debug the program.

v) Run the program, input data, and get the results.

**1.4 DESIGN OF ALGORITHMS**

The first step in the program development is to devise and describe a precise plan of what you want the computer to do. This plan, expressed as a sequence of operation, is called an algorithm. An algorithm is just an outline or idea behind a program..

**1.4.1 Definition**

An algorithm is a finite set of steps defining the solution of a particular problem. An algorithm is expressed in pseudo code-something resembling C language or Pascal, but with some statements in English rather than within the programming language. Developing an efficient algorithms one which is capable of giving the solution to the problem by using minimum resources of the system such as memory and processor’s time. Algorithm is a language independent, well structured and detailed. It will enable the programmer to translate into a computer program using any high-level language.

***Problem solving* 3**

**1.4.2 Features of Algorithm**

Following features should be present in an algorithm:

**Proper understanding of the problem** 

For designing an efficient algorithm, the expectations from the algorithm should be clearly defined so that the person developing the algorithm can understand the expectations from is. This is normally the outcome of the problem definition phase.

**Use of procedures/functions to emphasize modularity** 

To assist the development, implementation and readability of the program, it is usually helpful to modularize (section) the program. Independent functions perform specific and well defined tasks. In applying modularization, is is important to watch that the process is not taken so far to a point at which the implementation becomes difficult to read because of fragmentation. The program then can be implemented as calls to the various procedures that will be needed in the final implementations.

**Choice of variable names** 

Proper variable names and constant names can make the program more meaningful and easier to understand. This practice tends to make the program more self-documenting. A clear definition of all variables and constants at the start of the procedure/algorithm can also be helpful. For example, it is better to use variable *day* for the day of the weeks, instead of the variable A or something else.

**Documentation of the program** 

Brief information about the segment of the code can be included in the program to facilitate debugging and providing information. A related p art of the documentation is the information that the programmer presents to the user during the execution of the program. Since the program is often to be used by persons who are unfamiliar with the working and input requirements of the program, proper documentation must be provided. That is, the program must specify what responses are required from the user. Care should also be taken to avoid ambiguities in these specifications also the program should “catch” incorrect responses to its requests and inform the user in an appropriate manner.

**1.4.3 Criteria to be followed by an algorithm**

The following is the criteria to be followed by an algorithm:

i) **Input:** There should be zero or more values which are to be supplied.

ii) **Output:** At least one result is to be produced.

iii) **Definiteness:** Each step must be clear and unambiguous.

iv) **Finiteness:** Each step must be sufficiently a basic that a person using only paper and pencil canin principle carry it out. In addition, not only each step isdefinite, it must also be feasible.

**Example 1.1:** An algorithm to find the area of a Circle of radius r.

**Inputs to the algorithm:**

Radius r of the Circle. 

**Expected output:** 

Area of the Circle

**Algorithm:**

Step1: Read\input the Radius r of the Circle

Step2: Area= PI\*r\*r // calculation of area

Step3: Print Area

**Example 1.2:** An algorithm to compute and display the sum of two numbers

**Inputs to the algorithm:** 

Two numbers

**Expected output:**

****Sum of two numbers

***Problem solving* 4**

**Algorithm:**

Step1: Read two numbers a and b

Step2: Calculate the sum of a and b and store it is sum

Step3: Print sum

**Example 1.3:** Ravi has to attend at least 70% of Practical Classes for C programming to be eligible to appear in the external examination. Maximum no. of practical classes allotted for the course is 50. He has attended 20 out of 30 classes held so far. Find at least how many more classes to be attended by Ravi to be eligible for appearing in Practical Examination.

**Inputs to the algorithm:**

****Minimum Percentage of Attendance required appearing for the external exams. Maximum Number of practical classes 

Number of Classes held so far.

Number of classes attended by Ravi so far. 

**Expected output:**

Number of classes to be attended by Ravi to get eligibility for appearing the external examination **Algorithm:** 

Step1: Read Minimum percentage of attendance required.

Step2: Read Maximum no. of practical classes in the course (P)

Step3: Read Classes already attended (Ca)

Step4: Read No. of classes conducted so far.(CT)

Step5: Find the no. of Classes to be attended byRavi (Ct= C\*P/100)

Step6: Print CM.

**Example 1.4:** An algorithm to convert temperature from Fahrenheit to Celsius

**Inputs to the algorithm:**

Temperature in Fahrenheit 

**Expected output:** 

Temperature in Celsius

**Algorithm:**

Step 1: Read Temperature in Fahrenheit F

Step 2: C= 5/9\*(F=32)

Step 3: Print Temperature in Celsius: C

Step 4: Stop.

**Example 1.5:** An algorithm to compute and print the average of a set of data values. **Inputs to the algorithm:**

List of data values 

**Expected Output:**

Average of the data values 

Step1: Set the sum of the data values and the count to zero

Step2: As long as the data values exist, add the next data value to the sum andadd 1 to the count.

Step3: To compute the average, divide the sum by the count.

Step4: Print average.

**Example 1.6:** An algorithm to calculate the factorial of a given number.

**Inputs to the algorithm:**

****An integer number

***Problem solving* 5**

**Expected Output:**

Factorial of the integer 

step1: Read the number n

step2: [initialize] i 1, fact 1

step3: Repeat steps 4 through 5 until i = n

step4: fact fact\*i

step5: i i +1

step6: Print fact

**Example 1.7:** An algorithm to check a given integer is prime or not

**Inputs to the algorithm:**

An integer number 

**Expected Output:**

Prime or not 

step1: Read the number n

step2: [initialize] i 2, flag 1

step3: Repeat steps 4 through 5 until i < num or flag = 0

step4: rem num mod i

step5: if rem = 0 then flag 0 else i i + 1

step6: If flag = 0 then Print number is not prime else print number is prime

**Example 1.8:** Ramshewar goes to market for buying some fruits and vegetables. He is having a currency of Rs 500 with him for marketing. From a shop he purchases 2.0 kg Apple priced Rs. 50.0 per kg, 1.5 kg Mango priced Rs.35.0 per kg, 2.5 kg Potato priced Rs.10.0 per kg, and 1.0 kg Tomato priced Rs.15 per kg. He gives the currency of Rs. 500 to the shopkeeper. Find out the amount shopkeeper will return to Ramshewar and also tell the total item purchased.

Before we write algorithm for solving above problem let we find out what the inputs to the algorithm are and what expected output is.

**Inputs to the algorithm are:**

****Quantity of different items purchased. 

Unit Price of each item.

Total amount given to the shopkeeper. 

**Expected output:**

Amount to be returned by shopkeeper after deducting total price of the purchased vegetables and fruits. 

**Algorithm:**

Step1: Total Cost=0;

Step2: Read Number of units of ith item purchased;

Spet3: Read unit price of ith item

Step4: cost of ith item (CI) = number of units \* unit price of ith item.

Step5: total cost = total cost +CI.

Step6: i= i+1;

Step7: if i<=4 goto step 2.

Step7: RefundAmount = GivenAmount-Total Cost

Step8: Print RefundAmount

**Example 1.9:** Print the Multiplication Table of N.

**Inputs to the algorithm are:**

•Number N

**Expected output:**

***Problem solving* 6**

Table of N 

**Algorithm:**

Step 1: I=1

Step 2: Read N

Step 3: If I <= 10 then print I\*N otherwise goto Step 6

Step 4: I =I+1

Step 5: repeat step3 and 4

Step 6: stop

**1.4.4 Top down Design**

Once we define a problem and have an idea of how to solve it, we can then use the powerful techniques for designing algorithms. Most of the problems are complex or large problems and to solve them we have to focus on to comprehend at one time, a very limited span of logic or instructions. A technique for algorithm design that tries to accommodate this human limitation is known as top-down design or stepwise refinement.

Top down design provides the way of handling the logical complexity and detail encountered in computer algorithm. It allows building solutions to problems in step by step. In this war specific complex details of the implementation are encountered only at the stage when sufficient ground work on the overall structure and relationship among the various parts of the problem.

Before the top down design can be applied to any problem, we must at least have the outlines of a solution. Sometimes this might demand a lengthy and creative investigation in to the problem while another time the problem description may in itself provide the necessary starting point for the top down design.

Top down design suggests taking the general statements about the solution one at a time, and then breaking them in to a more precise subtask/sub-problem. This sub-problem should more accurately describe how the final goal can be reached. The process of repeatedly breaking a task down in to sub task in the smaller sub tasks must continue until the sub-problem can be implemented as the program statement .With each splitting it is essential to define how sub problem interact with each other. In this way the overall structure of the solution problem can be maintained. Preservation of the overall structure is important for making the algorithm comprehensible and also for making it possible to prove the correctness of the solution.

Complex or

largeproblem

Sub Task 1

Sub Task Sub Task 3 2

Sub Task N

Solution 1

Function …1( )

Solution 2

Function …2( )

Solution 3

Function

…3( )

Complex

solution

Solution 4

Function …4( )

Fig 1.1. Schematic break down of problem into subtasks as employed in top down designs

***Problem solving* 7**

**1.5 FLOW CHARTS**

“Flow charts are pictorial representation of an algorithm”. It shows the flow of operations in pictorial form.

**1.5.1 Flowchart Symbols**

For drawing flow chart standard symbols are used. These symbols are given in table.

| **Symbols** | **Meaning/Used for** |
| --- | --- |
|  | Start or end of the program |
|  | Used for writing steps of operations/action or processing function of a program |
|  | Input or output operation |
|  | Decision making and branching operations |
|  | Connector or joining of two parts in a flowchart |
|  | Flow line used for showing flow of data |
|  | Magnetic Tape used for secondary  storage/Backup |
|  | Magnetic Disk used for secondary  storage/Backup |

Fig1.2.Flow chart Symbols

**1.5.2 Significance of flowchart**

i) A flowchart is a diagrammatic representation of algorithm.

ii) A flow chart clearly illustrates the sequence of operations to be performed for getting the solution of a problem.

iii)For simple problems flow charts may not be very useful but for complex and large problems flow charts are very helpful in understanding the logic of the problem. iv) Flowcharts are used as a link of communication between programmers and clients for whomthe program to be developed.

v) If you are having a flowchart for your program then you can use it in explaining the programto others.

vi) Once the flowchart is drawn, it becomes easy to write the computer program.

***Problem solving* 8**

Flowchartscan be used for preparing a better documentation of a complex problem.

**1.5.3 Guidelines for drawing a flowchart**

i) Flow charts are drawn using slandered flowchart symbols.

ii) While drawing flowchart some guideline to be followed.

**Below are some guidelines for drawing flowchart:**

i) First of all list all necessary requirements in a logical order.

ii) The flowchart should be clear and easy to understand. There should not beany ambiguity in understanding the flowchart. For doing this it is necessary to have all the steps and operation very simple.

iii)Usually direction of the flow of data /procedure in the system should be from left to right or top to bottom.

iv) Only one flow line should come out from a process symbol.

v) When decision symbol is used only one flow line should enter toit (decision symbol), but there may be two or three flow lines coming out of the decision symbol, one for each possible answer.

A < B A>B

**Compare**

**A& B**

A=B

vi) In a flowchart only one flow line should come to the end symbol.

**END**

vii) While writing steps inside the processing symbol, steps should be brief and if necessary, you can use the annotation symbol to describe data or processing steps more clearly.

**P=A\*R**

Where P = Price, A = Amount, and R = Rate

viii) In the case of complex flowchart connector symbols to be used for reducing the number of flow lines in the flowchart.

ix) Intersection of flow lines should be avoided to make a flowchart more effective and

***Problem solving* 9**

for better way of communication.

x) A flowchart must have a logical start and end*.*

xi) Once a flowchart is drawn its validity should be tested by passing through it with a simple set of test data.

**start**

**Example1.10:** Draw a flowchart to find the simple interest

**accept p,t,r**

**i=ptr/100;**

**print i;**

**stop**

**Example1.11:** Draw a flowchart to find the sum of first 50 natural numbers **start**

**Sum=0**

**N=0**

N=**N+1**

**sum=sum+N**

**is**

**N=50?**

**yes**

**print sum**

**stop**

***Problem solving* 10**

**Example1.12:** Draw a flowchart to find the largest of three numbers A, B, and C. **start**

**Read A B C**

**Yes**

**is**

**B>C? B>C**

**No**

**No**

**is**

**A>B?**

**Yes**

**No**

**is**

**A>C?**

**Yes**

**print C print A**

**print B**

**stop**

**print c**

**Example1.13:** Draw a flowchart for computing factorial N (N!) Where N! = 1? 2? 3...N. **start**

**Read N**

**M=1**

**F=1**

**F=F\*M**

**NO**

**M=M+1**

**Is M=N?**

**YES**

**Print F**

**stop**

***Problem solving* 11**

**Example1.14:** Draw a flow chart to find the roots of a quadratic equation **start**

**Read A B**

**C**

**D=B\*B-4\*A\*C**

**False True**

**IS**

**D>0**

**True False**

**IS**

**D>0**

**ROOT1=(-B+sqrt(D))/(2\*A) ROOT2=(-B+sqrt(D))/(2\*A**

**PRINT “REAL**

**RP=-B/2\*A**

**IP=SQRT(-D)/2\*A**

**PRINT “ IMAGINARY ROOTS”**

**ROOT1=-B/2\*A ROOT2=-B/2\*A**

**PRINT “REAL AND EQUAL”**

**AND**

**DISTINCT”**

**PRINT**

**ROOT1,ROOT2**

**PRINT RP,IP PRINT ROOT1,ROOT2**

**STOP**

***Problem solving* 12**

**Example1.15:** Draw a flowchart to print biggest number from given list of numbers

**start**

**Read N,X**

**BIG=X**

**COUNT=1**

**IF**

**COUNT<N**

**Read X**

**IF X>BIG**

**BIG=X**

**COUNT=COUNT+1**

**Print BIG**

***Problem solving* 13**

**Example1.16:** Draw a flowchart to display the given number is prime or not **START**

**READ N**

K=√N

I=2; FLAG=1

R=remainder of(N/I)

IS

R=0

I=I+1 IS I<=K

IS

FLAG= 0

FLAG=0

PRINT PRIME PRINT NOT PRIME

***Problem solving* 14**

**Example1.17:** Draw a flowchart to generate first n elements of Fibonacci series. **START**

**READ N**

**FIB1=1**

**FIB2=1**

**PRINT FIB1, FIB2**

**COUNT=2**

**Yes**

**FIB3=FIB1=+FIB2**

**No**

**PRINT FIB3**

**COUNT=COUNT+1**

**IS COUNT<N**

**FIB1=FIB2**

**FIB2=FIB3STOP**

**1.5.4 Advantages of using Flowcharts**

As we discussed flow chart is used for representing algorithm in pictorial form. This pictorial representation of a solution/system is having many advantages. These advantages are as follows:

**i) Communication**: A Flowchart can be used as a better way of communication of the logic of a system and steps involve in the Solution, to all concerned

particularly to the client of system.

**ii) Effective analysis:** A flowchart of a problem can be used for effective analysis of the problem.

**iii) Documentation of Program/System**: Program flowcharts are a vital part of good

***Problem solving* 15**

program documentation. Program document is used for various purposes like

knowing the components in the program, complexity of the program etc.

**iv) Efficient Program Maintenance:** Once a program is developed and becomes operational it needs time to time maintenance. With help of flowchart maintenance become easier.

**v) Coding of the Program:** Any design of solution of a problem is finally converted into computer program. Writing code referring the flowchart of the solution become easy.

**1.5.5 Limitations of using Flowcharts**

• **Complexity of Logic**: If program logic is complex then flowchart of the program becomes complicated.

• **Alterations and Modifications in Logic**: any alterations in the program logic may require redrawing of flowchart completely.

• **Reuse is Not Possible**: As the flowchart symbols cannot be typed, always reproduction of flowchart symbols is required.

**1.6 LAUNGAGE**

A language is a mode of communication between two people. It is necessary for those two people to understand the language in order to communicate. But even if the two people do not understand the same language, a translator can help to convert one language to the other. Similarly we need a language to communicate with the computer. A translator is also needed to convert from user’s form to computers form. Like other language, a computer language also follows a particular grammar known as the syntax.

**1.7 PROGRAM AND A PROGRAMMING LANGUAGE**

To perform any task using a computer it is necessary to give a set of instructions to it. This set of instructions arranged in a logical sequence is called a Program. In practice it is necessary to express a program using a programming language. A procedure expressed in a Programming language is known as a Computer Program.

Programming languages can be divided into three categories.

**1.7.1 Machine level Language**

i) The language whose design is governed by the circuitry and the structure of the machine is known as machine language.

ii) This language is difficult to learn and use.

iii) It is specific to a given computer and is different for different computers

i.e this languageis machine-dependent.

iv)This language has been designed to give a better machine efficiency, i.e. faster program execution.

v) Machine language is also known as low level language.

**1.7.2 Assembly language**

i) We code the assembly language program in the form of mnemonics.

ii) Every machine provides a different set of mnemonics to be used for that only depending upon the processor that the machine is using. Hence, this language is also machine-dependent.

iii) Machine language is also known as low level language.

**1.7.3 High level language**

i) These languages have been designed to give a better programming efficiency, i.e

***Problem solving* 16**

faster program development.

ii) Every high level language follows a precise set of rules. They are developed to allow application programs to be run in a variety of computers.

iii) These languages are machine independent.

iv) Languages falling in this category are FORTRAN, BASIC, and PASCAL etc. v) These languages are easy to learn and programs may be written in these languages with much less effort.

vi) However, the computer cannot understand them and they need to be translated into machine language with the help of other programs known as compilers or interpreters.

**SUMMARY**

Computer is a powerful problem solving tool**.** Problem-solving skills are recognized as an integral component of computer programming. A problem solving technique follows certain steps in finding the solution to a problem. An algorithm is a finite set of steps defining the solution of a particular problem. Flow chart is pictorial representation of an algorithm. It shows the flow of operations in pictorial form. For drawing flow chart standard symbols are used. The START and STOP are

represented by an ellipse like figure , decision by the rhombus like figure the process by rectangle and input/ output by parallelograms . . Lines and arrows connect these blocks. A program is sequence of instructions and the process of writing program is called programming. Programming languages can be divided into three categories: machine language, assembly language and high level language. High level languages are easy to use while machine and assembly languages are complex. Therefore, writing programs in machine and assembly languages is difficult and time consuming.

**EXERCISES**

**Multiple Choice Questions**

1.1 C is --------- language

a) machine language b) assembly language c) high level language d) fourth generation language 1.2 Which of the following language is also called binary language?

a)high level language b)assembly language c)machine language d)none of these 1.3 Which of the following language is machine independent?

a)high level language b)assembly language c)machine language d)none of these 1.4 Which of the following is not high level language?

a) java b)Pascal c)COBOL d)oracle

***Problem solving* 17**

**Compressive Questions**

1.1 What is algorithm and explain it briefly?

1.2 What is flowchart? What are different symbols used for design of flow chart? 1.3 Design an algorithm and draw flowchart to find area and circumference of a circle 1.4 Design an algorithm and draw flowchart to find area and circumference of a rectangle 1.5 Design an algorithm and draw flowchart to check the given character is vowel or not 1.6 Given the marks obtained by the students, maximum marks and pass marks in three subjects, design

an algorithm and draw flowchart to find whether the student passed or not, if the student passes determine the percentage marks and grade.

The grade is determined as follows:

i) percentage marks >=80 grade is A

ii) percentage marks >=70 and <80 grade is B

iii) percentage marks >=60 and <70 grade is C

iv) percentage marks >=50 and <60 grade is D

v) percentage marks <50 grade is F

1.7 Design an algorithm and draw flowchart for converting temperature from Fahrenheit to centigrade and vice-versa

1.8 Design an algorithm and draw flowchart to find roots of a quadratic equation 1.9 Design an algorithm and draw flowchart to find the nature of the triangle when sides are given 1.10 Design an algorithm and draw flowchart to find sum of n numbers.

1.11 Design an algorithm and draw flowchart to check given number is prime or not 1.12 Design an algorithm and draw flowchart to check given number is strong or not 1.13 Design an algorithm and draw flowchart to check given number is palindrome or not 1.14 Design an algorithm and draw flowchart to check given number is perfect or not 1.15 Design an algorithm and draw flowchart to check given number is Armstrong or not 1.16 Design an algorithm and draw flowchart to find the smallest number in a given list of numbers 1.17 Design an algorithm and draw flowchart to find first and second largest numbers in a given list

***Problem solving* 18**