1. Define number system? What are the different types of number systems?

A.

Number System: It is a way to represent the numbers in particular formats

There are 4 number systems:

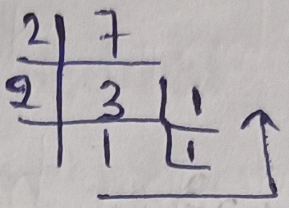
Binary number system.

Octal number system.

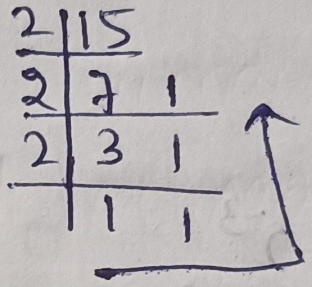
Decimal number system.

Hexa decimal number system.

Ex $(7)_{10} = (111)_2$

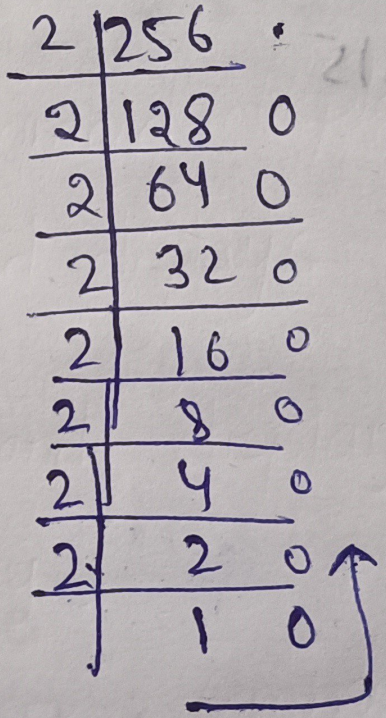


$(15)_{10} = (1111)_2$



$(256)_{10}$

$= (100000000)_2$



Decimal to Octal

Given number is divided with bottom to top

2. Define Algorithm and what are the properties of an algorithm?

A.

An algorithm is a finite set of steps defining the solution of a particular problem. .

The characteristics of an algorithm are as follows:

a. Input
b. Output
c. Finiteness
d. Definiteness
e. Effectiveness

Two operator characteristics determine how operands group with operators: *precedence* and *associativity*. Precedence is the priority for grouping different types of operators with their operands. Associativity is the left-to-right or right-to-left order for grouping operands to operators that have the same precedence. An operator's precedence is meaningful only if other operators with higher or lower precedence are present. Expressions with higher-precedence operators are evaluated first. The grouping of operands can be forced by using parentheses.

For example, in the following statements, the value of 5 is assigned to both a and b because of the right-to-left associativity of the = operator. The value of c is assigned to b first, and then the value of b is assigned to a.

```
b = 9;
c = 5;
a = b = c;
```

Because the order of subexpression evaluation is not specified, you can explicitly force the grouping of operands with operators

3. What is conditional operator or ternary operator?

A.

The Conditional Operator in C, also called a Ternary operator, is one of the Operators, which used in the decision-making process. The C Programming Conditional Operator returns the statement depends upon the given expression result.

syntax: Test_expression ? statement1: statement2

What is Type Casting in C?

Type casting is the process in which the compiler automatically converts one data type in a program to another one. Type conversion is another name for type casting. For instance, if a programmer wants to store a long variable value into some simple integer in a program, then they can type cast this long into the int. Thus, the method of type casting lets users convert the values present in any data type into another data type with the help of the cast operator, like:

(type_name) expression

Let us take a look at the following example to understand how we can utilise the cast operator for dividing an integer variable with another one by performing it as an operation of floating-point type:

```
#include <stdio.h>

main() {

int total = 17, values = 5;

double average;

average = (double) total / values;

printf("The average of all the values available with us is : %f\n", average );

}
```

The compilation and execution of the code mentioned here would generate the following

: 3.400000

You must note here that the precedence of the cast operator is much higher than that of the division operator. Thus, the program would first convert the value of *total* into the *double* type. After this, it will finally divide this value with the help of count yielding of the double value.

As a matter of fact, the process of type conversion or type casting can be very implicit. It means that the compiler is capable of performing it automatically. Conversely, we can specify the data type explicitly using the cast operator. Both the methods are okay, but using the cast operator whenever in need is considered a better programming practice (explicit).

Syntax for Type Casting in C

```
int val_1;

float val_2;

// Body of program

val_2 = (float) val_1; // type casting of the values
```

Types of Type Casting in C

The process of type casting can be perform
in two major types in a C program. These

- Implicit
- Explicit

- Replacing symbolic operation code by machine operation codes
- Reserving storage for the instructions and data
- Translating constants into their machine representation

- **Compiler** The compiler is a computer program that translates the source code written in a high-level language into the corresponding *object code* of the low-level language. This translation process is called *compilation*. The entire high-level program is converted into the executable machine code file. A program that translates from a low-level language to a high-level one is a decompiler. Compiled languages include COBOL, FORTRAN, C, C++, etc.

In 1952, Grace Hopper wrote the first compiler for the A-0 programming language. In 1957, John Backus at IBM introduced the first complete compiler. With the increasing complexity of computer architectures and expanding functionality supported by newer programming languages, compilers have become more

and more complex. Though early compilers were written in assembly languages, nowadays it has become common practice to implement a compiler in the language it compiles. Compilers are also classified as *single-pass compilers* and *multi-pass compilers*. Though single-pass compilers are generally faster than multi-pass compilers, for sophisticated optimization, multi-pass assemblers are required to generate high-quality code.

- **Interpreter** The interpreter is a translation program that converts each high-level program statement into the corresponding machine code. This translation process is carried out just before the program statement is executed. Instead of the entire program, one statement at a time is translated and executed immediately. The commonly used interpreted language are BASIC and PERL. Although, interpreters are easier to create as compared to compilers, the compiled languages can be executed more efficiently and are faster.

Linkers

Most of the high-level languages allow the developer to develop a large program containing multiple modules.

ures

run on the

a numeral
alues using
s bits.
ic code that
characters,

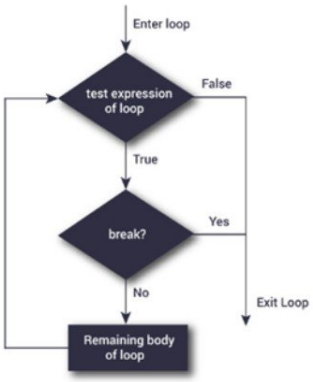
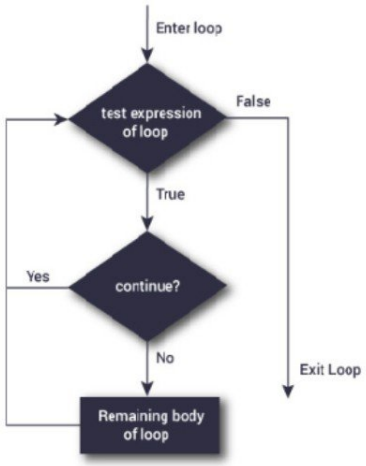
- **Compiler**: It is a computer program that translates the source code written in a high-level language into the corresponding object code of the low-level language.
- **Interpreter**: It is a translation program that converts each high-level program statement into the corresponding machine code.

UNIT

2 Int

1

3.2k views

Break	Continue
When break is encountered the switch or loop execution is immediately stopped.	When continue is encountered, the statements after it are skipped and the loop control jump to next iteration.
break statement is used in switch and loops.	continue statement is used in loops only.
Flowchart: 	Flowchart: 

0

1.5k views

```
#include<stdio.h>
main()
{
    int i;
    for(i=0;i<5;++i)
   {
         if(i==3)
         break;
         printf("%d ",i);
     }

}

Output:
0 1 2
```

Continue Example:

```
#include<stdio.h>
main()
{
    int i;
    for(i=0;i<5;++i)
    {
          if(i==3)
          continue;
          printf("%d ",i);
      }

}

Output:
```

Application

- Computer Languages :

→ low level language / Machine language / Binary language

→ Assembly level language / Symbolic language / Middle-level language

→ high-level language

(i) Machine language

→ 0's and 1's

→ Machine dependent

→ programs written in the form of 0's and 1's

→ Machine readable

(ii) Assembly language

→ Mnemonics or Symbols used in writing programm

→ Very difficult to understand by human

→ Machine dependant

→ Ex:- ADD, SUB, MUL, DIV... etc.

(iii) High level language

→ It is easier to learn

→ understanble by humans

→ needs a translator called "compiler"

→ error recovery and correction is very easy.

12) Explain the

Part B

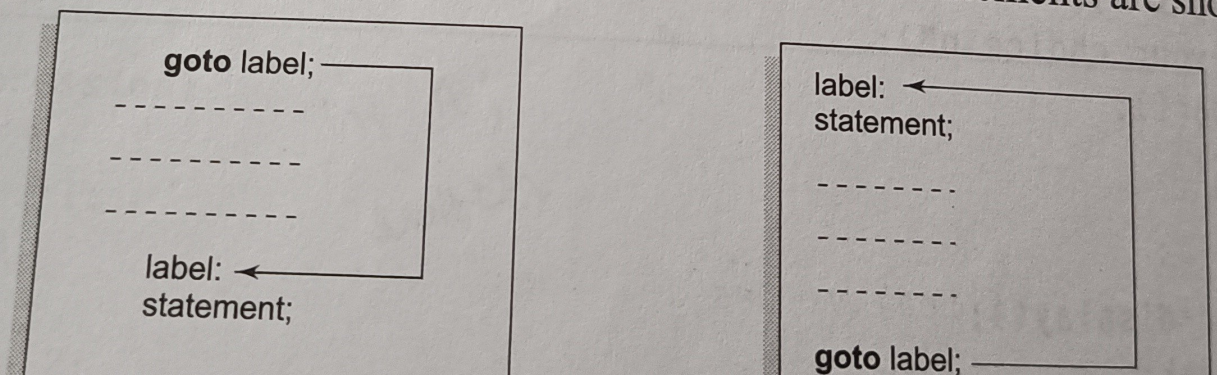
1) Explain

2) differentiate

2.21.3 The goto Statement

So far we have discussed ways of controlling the flow of execution based on certain specified conditions. Like many other languages, C supports the `goto` statement to branch unconditionally from one point to another in the program. Although it may not be essential to use the `goto` statement in a highly structured language like C, there may be occasions when the use of `goto` might be desirable.

The `goto` requires a label in order to identify the place where the branch is to be made. A label is any valid variable name, and must be followed by a colon. The label is placed immediately before the statement where the control is to be transferred. The general forms of `goto` and label statements are shown in Fig. 2.58:



operator is required.

2.19.7 Bitwise Operators

C has a distinction of supporting special operators known as bitwise operators for manipulation of data at bit level. These operators are used for testing the bits, or shifting them right or left. Bitwise operators may not be applied to float or double. Table 4.20 provides a list the bitwise operators and their meanings.

TABLE 2.20 Bitwise operators

Operator	Meaning
&	bitwise AND
|	bitwise OR
^	bitwise exclusive OR
<<	shift left
>>	shift right

2.19.8 Special Operators

C supports some special operators of interest such as comma operator, sizeof operator, pointer operators (& and *) and member selection operators (. and –>).

aracterisitics of a computer

peed

torage capacity

ccuracy

liability

satility

ligence

→ Write a C program using Bitwise Operator

```
# include <stdio.h>
  main( )
{
int a,b,c;
  a=5;
  b=6;
  c= a&b;
{
Print f ("%d & %d = %d \n", a,b,c);
Print f ("%d \n", c);
c=a/b
Print f ("%d %/ %d = %d \n", a,b,c);
c= ~a;
Print f ("~ %d = %d \n", a, c);
c= a^b;
Print f ("%d\n", c);
c= a<< 1;
Print f ("%d \n", c);
c= b>>2;
Print f ("%d \n", c);
}
```

output : 4
-6
3
10
1

Write a 'C' program using bitwise operator

```
# include <stdio.h>
```

- For direct interaction with the hardware
- In extremely high-security situations where complete control over the environment is required
- To maximize the use of limited resources, in a system with severe resource constraints

1.3.3 High-Level Languages

High level languages further simplified programming tasks by reducing the number of computer operation details that had to be specified. High level languages like COBOL, Pascal, FORTRAN, and C are more abstract, easier to use, and more portable across platforms, as compared to low-level programming languages. Instead of dealing with registers, memory addresses and call stacks, a programmer can concentrate more on the logic to solve the problem with help of variables, arrays or Boolean expressions. For example, consider the following assembly language code:

```
LOAD A
ADD B
```

STORE C

Using FORTRAN, the above code can be represented as:

$C = A + B$

The above high-level language code is executed by translating it into the corresponding machine language code with the help of a compiler or interpreter.

High-level languages can be classified into the following three categories:

- Procedure-oriented languages (third generation)
- Problem-oriented languages (fourth generation)
- Natural languages (fifth generation)

1.3.4 Procedure-oriented Languages

High-level languages designed to solve general-purpose problems are called *procedural languages* or *third-generation languages*. These include BASIC, COBOL, FORTRAN, C, C++, and JAVA, which are designed to express the logic and procedure of a problem. Although, the syntax of these programming languages

lines more in the examples that follow.

Let us now look at the printf( ) function, the only executable statement of the program, as show

```
printf("I see, I remember");
```

printf is a predefined standard C function for printing output. Predefined means that it is a functi already been written and compiled, and linked together with our program at the time of linking. function causes everything between the starting and the ending quotation marks to be printed out. I the output will be:

```
I see, I remember
```

Note that the print line ends with a semicolon. Every statement in C should end with a semicolo Suppose we want to print the above quotation in two lines as

- **Formatted output**: It refers to the generated output that has been arranged in a particular format.
- **printf**: It is a function used to print and display output of a program.
- **scanf**: It is a function used to read values entered by the user upon execution of a program.
- **Operator**: It is a symbol that tells the computer to perform certain mathematical or logical

3. What are the different types of operators used in C language?

A.

Operator:

Operator in C is a symbol that tells the computer to perform mathematical or logical manipulation on data.

Different operators are:

1. Arithmetic Operators (+, -, *, /, %)
2. Increment and Decrement Operators (++, --)
3. Assignment Operators (=, +=, -=, *=, /=, %=)

4. Relational Operators (==,>,<,!=,>=,<=)
5. Logical Operators (&&,||,!)
6. Conditional Operators (?:)
7. Bitwise Operators(&,|,~,^,<<,>>)
8. Special Operators (comma operator, sizeof operator)

Examples:

//arithmetic operators

c=a+b;

c=a-b;

c=a*b;

c=a/b;

//Increment and decrement operators

c++;

c--;

++c;

--c;

//Assignment operators

a=b;

a+=b;

a-=b;

a*=b;

//Relational Operators

4==3 returns 0

4>3 returns 1

4<3 returns 0

4!=3 returns 1

4>=3 returns 1

4<=3 returns 0

//Logical operators

if c = 5 and d = 2 then, expression ((c == 5) && (d > 5)) equals to 0.

if c = 5 and d = 2 then, expression ((c == 5) || (d > 5)) equals to 1.

if c = 5 then, expression ! (c == 5) equals to 0.

17. Differentiate while and do-while statements.

A.

while loop	**do while loop**
It is a entry control loop	It is a exit control loop
Statements will be executed 0 or more times	Statements will be executed 1 or more times
Condition will be checked first then statements will be executed.	Statements will be executed first then the condition will be checked.
If there is a single statement, brackets are not required.	brackets are always required.

18. What are the selection statements?

A.

The statements which are used to place the conditions in a c program to control the flow of execution of a c program are known as selection statements.

In c language the following are the selection statements

- simple if
- nested if
- if else
- else if ladder

```
1 ; counter <=100; counter ++)

er %. 2==0)

Print f ("%. d\n", counter);
}
}
return 0;
}
```

Output:

2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24,
26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46,
48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68,
70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90,
94, 96, 98, 100.

Write a C program to display prime numbers from 1 to 100 using functions.

```
# include <stdio.h>
void prime()
{
int i=2, j, p;
while (i<=100)
{
 p=1;
 for (j=2 ; j<i ; j++)
 {
 if (i%. j==0)
 {
  P=0;
 }
}
}
 if (p)
{
 print f ("\t %. d \n", i);
```

explain different type of operators used in functions
differentiate while and do-while (5)
what are selection statements? with example.
Write

```
}
i++;
}
}
int main()
{
Printf ("Prime numbers between 1 to 100 \n");
Printf ("-------\n");
 Prime();
 return 0;
}
```

Output:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

Write a C program that uses are recursive function to solve the tower of Hanoi problem.

```
#include <stdio.h>
Void towers (int, char, char, char);
int main()
{
int num;
Printf ("Enter the number of disks");
Scanf ("%d", &num);
Printf ("The sequence of move involved in the pow
          of Hanoi are :\n");
towers(num, 'A', 'C', 'B');
```

processed directly by

Octal
Decimal
Hexa De

• Number
1. Any

the help of an
steps in details (1.18)
example (2.67)
(2.90) (2.99)

```
    }
    i++;
    }
}
int main()
{
Printf ("Prime numbers between 1 to 100 \n");
Printf ("-------\n");
  Prime();
  return 0;
}
```

Output :-

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

```
1 2 3
ROOTS ARE IMAGINARY
```

Fig. 2.46 *Solution of a quadratic equation*

The term (b2–4ac) is called the discriminant. If the discriminant is less than zero, its square roots cannot be evaluated. In such cases, the roots are said to be imaginary numbers and the program outputs an appropriate message.

2.21 DECISION MAKING AND BRANCHING

We have seen that a C program is a set of statements that are normally executed sequentially in the order in which they appear. This happens when no options or no repetitions of certain calculations are necessary.

However, in practice, we have a number of situations where we may have to change the order of execution of statements based on certain conditions, or repeat a group of statements until certain specified conditions are met. This involves a kind of decision making to see whether a particular condition has occurred or not and then direct the computer to execute certain statements accordingly.

C language possesses such decision-making capabilities by supporting the following statements:

- if statement
- switch statement
- goto statement

These statements are popularly known as decision-making statements. Since these statements 'control' the flow of execution, they are also known as control statements. We have already used some of these statements in the earlier examples. Here, we will discuss their features, capabilities and applications in detail.

2.21.1 Decision Making with if Statement

The if statement is a powerful decision-making statement and is used to control th execution of

1.5 CREATING AND RUNNING PROGRAMS

A programmer should adopt standard methodologies and approaches to program development. Software or *program-development life cycle* is one such standard methodology that is applicable to all types of program development scenarios. It comprises of a number of interlinked phases with each phase serving a definite purpose. We will study the program development life cycle in more detail in the next section.

1.5.1 Structured Programming

Another important program development approach is structured programming, which is a subset of one of the key programming paradigms, i.e., procedural programming.

It helps in making a program easily understandable and debuggable. A program that is not based on the structured programming approach is very difficult to maintain, debug and understand.

Structured programming approach mainly focuses on the order of execution of the statements within a program. It suggests the use of sequential execution of statements in a program. Thus, structured programming approach suggests the use of mainly three types of control structures—sequential, repetitive and selective.

Further, it suggests avoiding the use of **goto**, **break** and **continue** statements in a program as all these are unconditional branch statements.

1.5.2 System Development Tools

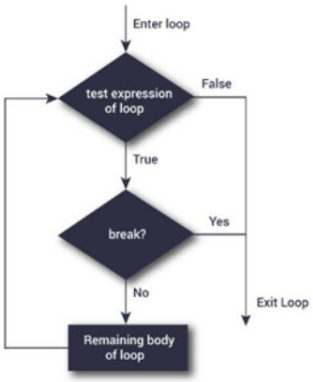
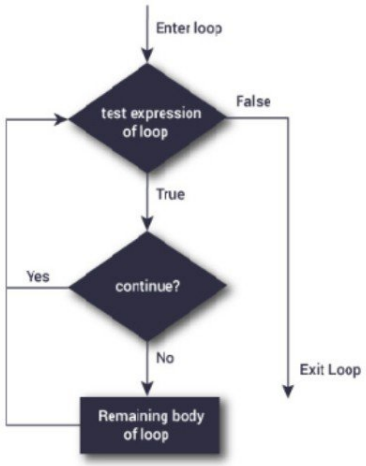
The successful development and execution of programs requires the usage of a number of tools. Some of these typical system development tools are

- Language translators
- Linkers
- Debuggers
- Editors

Language Translators

1

3.2k views

Break	Continue
When break is encountered the switch or loop execution is immediately stopped.	When continue is encountered, the statements after it are skipped and the loop control jump to next iteration.
break statement is used in switch and loops.	continue statement is used in loops only.
Flowchart: 	Flowchart: 

0

1.5k views

```
#include<stdio.h>
main()
{
    int i;
    for(i=0;i<5;++i)
   {
         if(i==3)
         break;
         printf("%d ",i);
     }

}

Output:
0 1 2
```

Continue Example:

```
#include<stdio.h>
main()
{
    int i;
    for(i=0;i<5;++i)
    {
          if(i==3)
          continue;
          printf("%d ",i);
      }

}

Output:
```

- **Binary file editor** It is used to edit the digital [illegible] and 1s.
- **HTML editor** It is used to edit the information included in the Web pages.
- **Source code editor** It is used to edit the source code of a program written in a programming language such as *C*, *C++* and Java.

1.5.3 Developing a Program

Developing a program refers to the process of writing the source code for the required application by following the syntax and the semantics of the chosen programming language. Syntax and semantics are the set of rules that a programmer needs to adhere while developing a program.

Before actually developing a program, the aim and the logic of the program should be very clear to the programmer. Therefore, the first stage in the development of a program is to carry out a detailed study of the program objectives. The objectives make the programmer aware of the purpose for which the program is being developed. After ascertaining the program objectives, the programmer needs to list down the set of steps to be followed for program development. This set of program development steps is called algorithm. The programmer may also use a graphical model known as flowchart to represent the steps defined in the program algorithm.

After the logic of the program has been developed either by an algorithm or a flowchart, the next step is to choose a programming language for actual development of the program code. There are a number of

Suppose w
for student
preparatio

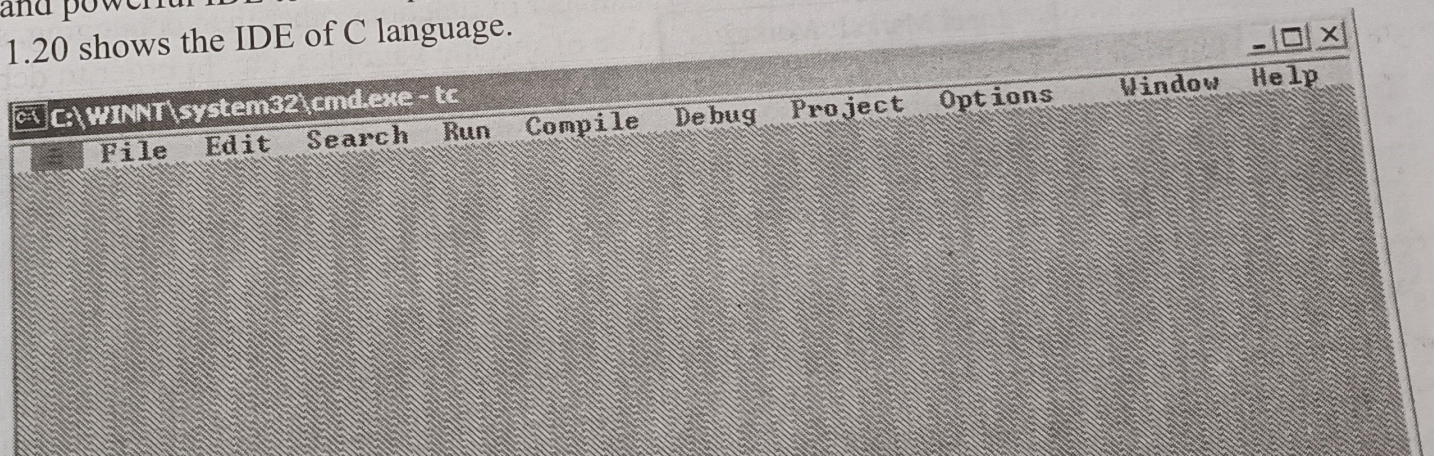
Step
Step
Step
Step
Ste
Ste
Ste

Figur

factors that should be taken into consideration while selecting the target programming language, such as performance and efficiency of the programming language, programmer's prior experience with the language, etc.

A programming language is typically bundled together with an IDE containing the necessary tools for developing, editing, running and debugging a computer program. For instance, Turbo C is provided with a strong and powerful IDE to develop, compile, debug and execute the programs.

Figure 1.20 shows the IDE of C language.



2.21.2 The switch Statement

We have seen that when one of the many alternatives is to be selected, we can use an `if` statement to control the selection. However, the complexity of such a program increases dramatically when the number of alternatives increases. The program becomes difficult to read and follow. At times, it may confuse even the person who designed it. Fortunately, C has a built-in multiway decision statement known as a `switch`. The `switch` statement tests the value of a given variable (or expression) against a list of case values and when a match is found, a block of statements associated with that case is executed. The general form of the switch statement is shown further:

```
switch (expression)
{
   case value-1:
      block-1
      break;
   case value-2:
      block-2
      break;
   ......
   ......
   default:
      default-block
      break;
}
statement-x;
```



```
printf(" TRAVEL
printf(" A Air T
printf(" T Trai
printf(" B Bus
printf(" X To
printf("\n Ent
character = ge
switch (chara
{
    case 'A'

    case '

    case

default :

}
```

• Characteristics of
→ Speed
→ Storage capacity

The switch statement is often used for menu selection. For example:

```
printf(" TRAVEL GUIDE\n\n");
printf(" A Air Timings\n" );
printf(" T Train Timings\n");
printf(" B Bus Service\n" );
printf(" X To skip\n" );
printf("\n Enter your choice\n");
character = getchar();
switch (character)
{
    case 'A' :
              air-display();
              break;
    case 'B' :
              bus-display();
              break;
    case 'T' :
              train-display();
              break;
default :
              printf(" No choice\n");
}
```

It is possible to nest the `switch` statements. That is, a `switch` may be part of a case statement. ANSI C permits 15 levels of nesting.

Here are some key rules for using `switch` statement:

- The `switch` expression must be an integral type.
- Case labels must be constants or constant expressions.
- Case labels must be unique. No two labels can have the same value.
- Case labels must end with semicolon.
- The `break` statement transfers the control out of the `switch` statement.
- The `break` statement is optional. That is, two or more case labels may belong to the same statements.
- The default label is optional. If present, it will be executed when the expression does not find a matching case label.
- There can be at most one default label.
- The default may be placed anywhere but usually placed at the end.
- It is permitted to nest switch statements.

advantages & dis
12) Explain the use
Part B

The test may be either to determine whether the loop has been repeated the specified number of times or to determine whether a particular condition has been met.

The C language provides for three constructs for performing loop operations. They are:

1. The `while` statement.
2. The `do` statement.
3. The `for` statement.

2.23.1 The while Statement

The simplest of all the looping structures in C is the `while` statement. We have used `while` in many of our earlier programs. The basic format of the `while` statement is

```
while (test condition)
{
body of the loop
}
```

The `while` is an entry-controlled loop statement. The test-condition is evaluated and if the condition is true, then the body of the loop is executed. After execution of the body, the test-condition is once again evaluated and if it is true, the body is executed once again. This process of repeated execution of the body continues until the test-condition finally becomes false and the control is transferred out of the loop. On exit, the program continues with the statement immediately after the body of the loop.

The body of the loop may have one or more statements. The braces are needed only if the body contains two or more statements. However, it is a good practice to use braces even if the body has only one statement.

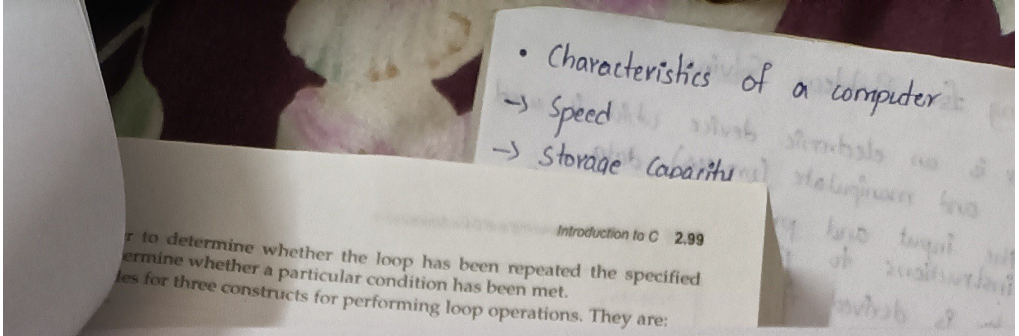
An example of `while` statement, which uses the keyboard input, is shown as here:

```
________
character = ' ' ;
while (character != 'Y')
      character = getchar();
xxxxxxx;
________
```

First the character is initialized to ' '. The `while` statement then begins by testing whether character is not equal to Y. Since the character was initialized to ' ', the test is true and the loop statement character = `getchar()`; is executed. Each time a letter is keyed in, the test is carried out and the loop statement is executed until the letter Y is pressed.

When Y is pressed, the condition becomes false because character equals Y, and the loop terminates, thus transferring the control to the statement `xxxxxxx`;.

Example 2.23 A program to evaluate the equation $y = x^n$, where n is a non-negative integer, is given in Fig. 2.63.



r to determine whether the loop has been repeated the specified
ermine whether a particular condition has been met.
es for three constructs for performing loop operations. They are:

if c = 5 and d = 2 then, expression ((c == 5) && (d > 5)) equals to 0.
if c = 5 and d = 2 then, expression ((c == 5) || (d > 5)) equals to 1.
if c = 5 then, expression ! (c == 5) equals to 0.

4. Briefly discuss about conditional or selection statements used in C language?

A.

conditional statements or selection statements requires that the programmer specifies one or more conditions to be evaluated or tested by the program along with statements to be executed, if the condition is determined to be true & Optionally, if the condition is false other statements will be executed.

if statement

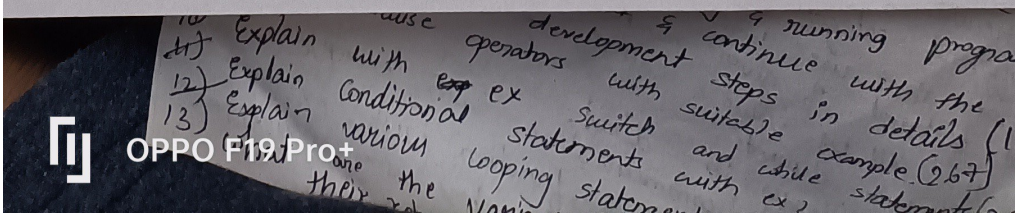
if statement is the simplest form of selection constructs. It is used to execute or skip a statement or statements by checking a condition. The condition is given as a relational expression. If the condition is true, the statement or set of statements after **if statement** is executed otherwise not executed.

Syntax:
The syntax of if statement is:

if (condition)
statement;

The above is for single statement. A set of statements can also be made conditional by writing the statements in braces { }. A set of statement is also called block of statements. The syntax for block of statements in if statements is:

f (condition)
{
tatement 1;



statement 2;

.

.

statement n;

}

if else statement

It executes one statement or block of statements when the condition is **true** but if condition is **false** then it will execute another statement or block of statements. In any situation, one block is executed & the other is skipped.

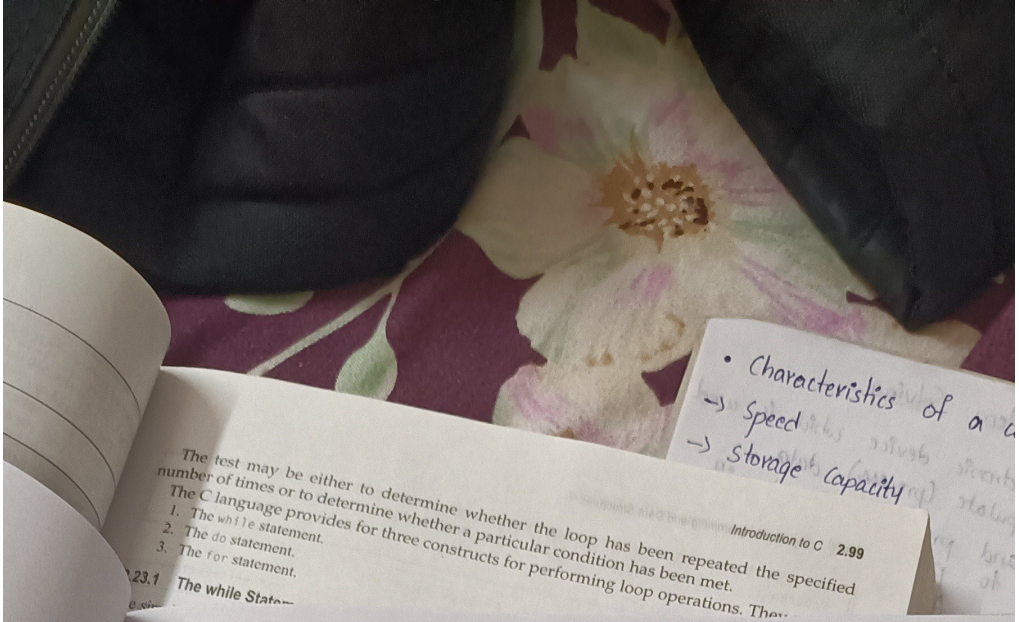
Syntax:

The syntax for single statement:

```
if (condition)
statement;

else

statement;
```

The syntax for Block of statements:

```
if (condition)
{
statement 1;
statement 2;
.
.
statement n;

}
else
{
```



• Characteristics of a c
→ Speed
→ Storage capacity

The test may be either to determine whether the loop has been repeated the specified number of times or to determine whether a particular condition has been met.
The C language provides for three constructs for performing loop operations. Ther
1. The while statement.
2. The do statement.
3. The for statement.

2.23.1 The while State

```
statement 1;
statement 2;
.
.
statement n;
}
```

Switch Statement

Switch statement is used for multiple choice or selection. It is used as a substitute of if-else statements. It is used when multiple choice are given & one choice is to be selected.

Syntax

```
switch(expression)
{
case const-1:
statements;
break;
case const-2:
statement;
break;
.
.
case const-n:
statement;
break;
```

```
default:

statements:

}
```

```
default:
statements:
}
```

5. Briefly discuss about looping or iterative statements?

A.

The looping statements in C language are also known as Repetition statements or Iterative statements. They are as follows.

1. while loop
2. do-while loop
3. for loop

Syntax:

1. while loop

```
Initialization…
while(condition)
{
        //body of the loop
loop update→ increment or decrement
}
```

2. do while loop

```
Initialization..
do
{
  //body of the loop
  loop update
```

```
}while(condition);
```

3. for loop

```
for(Initialization ;condition ; loop update)
{
        //body of the loop
}
```

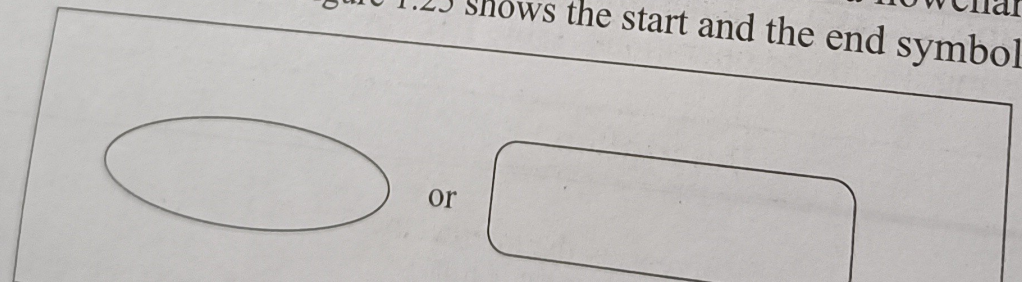
```
Step 8 - Set left = left + 1 and right = right + 1
Step 9 - If chk='t' goto Step 10 else goto Step 11
Step 10 - Display "The string is a palindrome" and goto Step 12
Step 11 - Display "The string is not a palindrome"
Step 12 - Stop
```

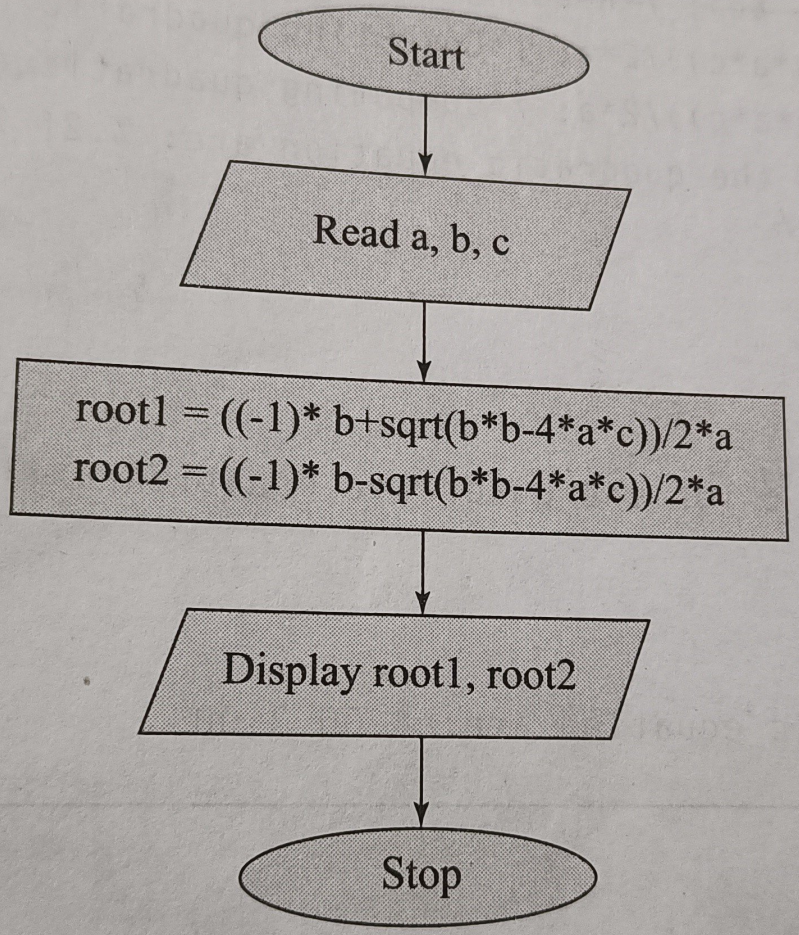
1.10 FLOWCHARTS

A flowchart can be defined as the pictorial representation of a process, which describes the sequence and flow of control and information within the process. The flow of information is represented inside the flowchart in a step-by-step form. This technique is mainly used for developing business workflows and solving problems using computers.

A flowchart uses different symbols for depicting different activities, which are performed at different stages of a process. The various symbols used in a flowchart are the following:

- **Start and end** It is represented by an oval or a rounded rectangle. It represents the starting and the ending of a process. Every process starts and ends at some point so a flowchart always contains one start as well as one end symbol. Figure 1.25 shows the start and the end symbols used in a flowchart.




Start
Read a, b, c
root1 = ((-1)* b+sqrt(b*b-4*a*c))/2*a
root2 = ((-1)* b-sqrt(b*b-4*a*c))/2*a
Display root1, root2
Stop