PPS (BCS-201)

Sessional Test-1

Solution

Section A

Answer 1 (a)

State desirable characteristics of an algorithm. Write an algorithm to calculate sum of digits of a 3 digits number entered by user.

Algorithm

An algorithm is a finite set of steps to solve a problem.

An algorithm must possess the following **characteristics**

1. Finiteness: An algorithm must terminate in a finite number of steps.

2. Input: Each algorithm must take zero, one or more quantities as input data

3. Output: Each algorithm must produce one or more output values.

4. Unambiguous: Each step of the algorithm must be precisely and **unambiguously (free from confusion)**

5. Feasible: The Algorithm must be simple, generic and practical, such that it can be executed upon with the available resources.

6. Language Independent: The algorithm should be written free from any programming language. It should be general which can be implemented in any programming language:

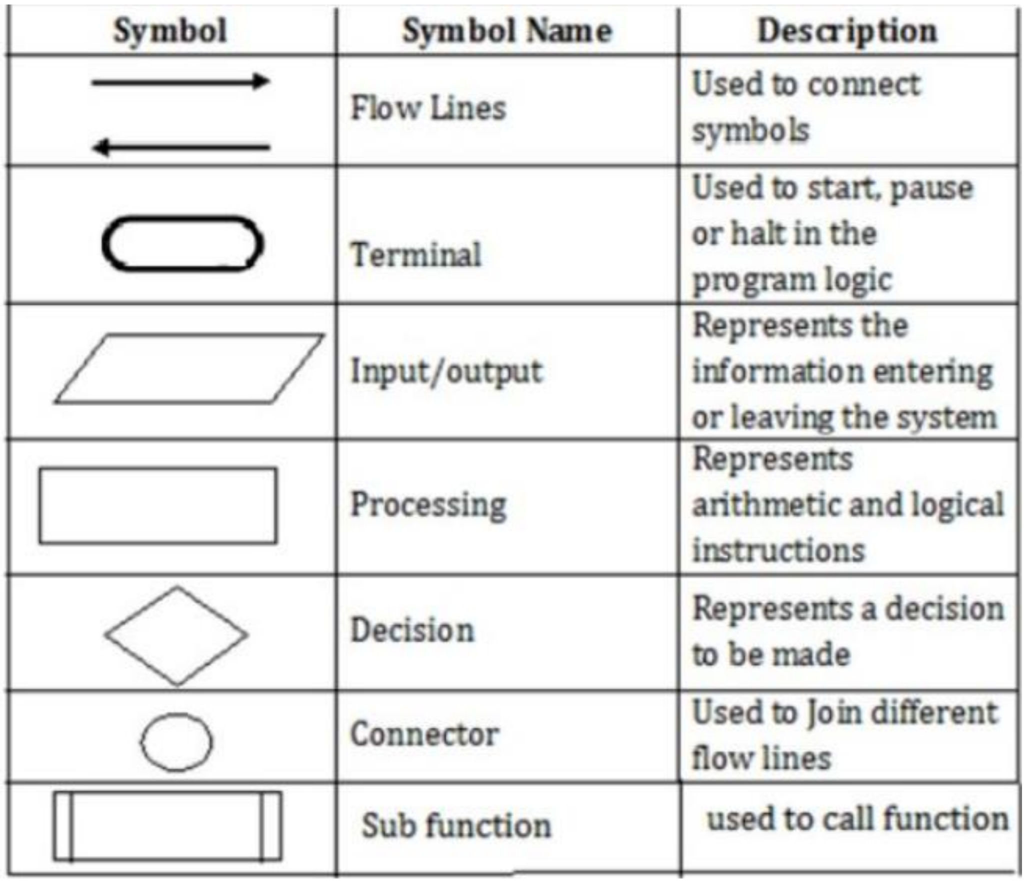
Write an algorithm to find the sum of 3 digit number entered by the user

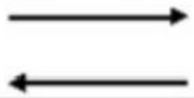
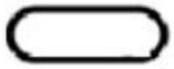
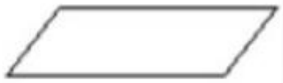
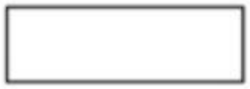
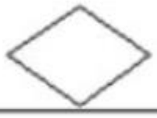
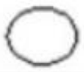
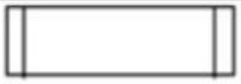
```
Step1: START/BEGIN
Step2:  Input three digit number in 'a' variable
Step3: b=a%10
Step 4: c=a/10
Step 5: d=c%10
Step 6: e=c/10
Step 7:  sum=b+d+e
Step 8: Print/Display sum
Step 9: End/Stop
```

Answer 1 (b)

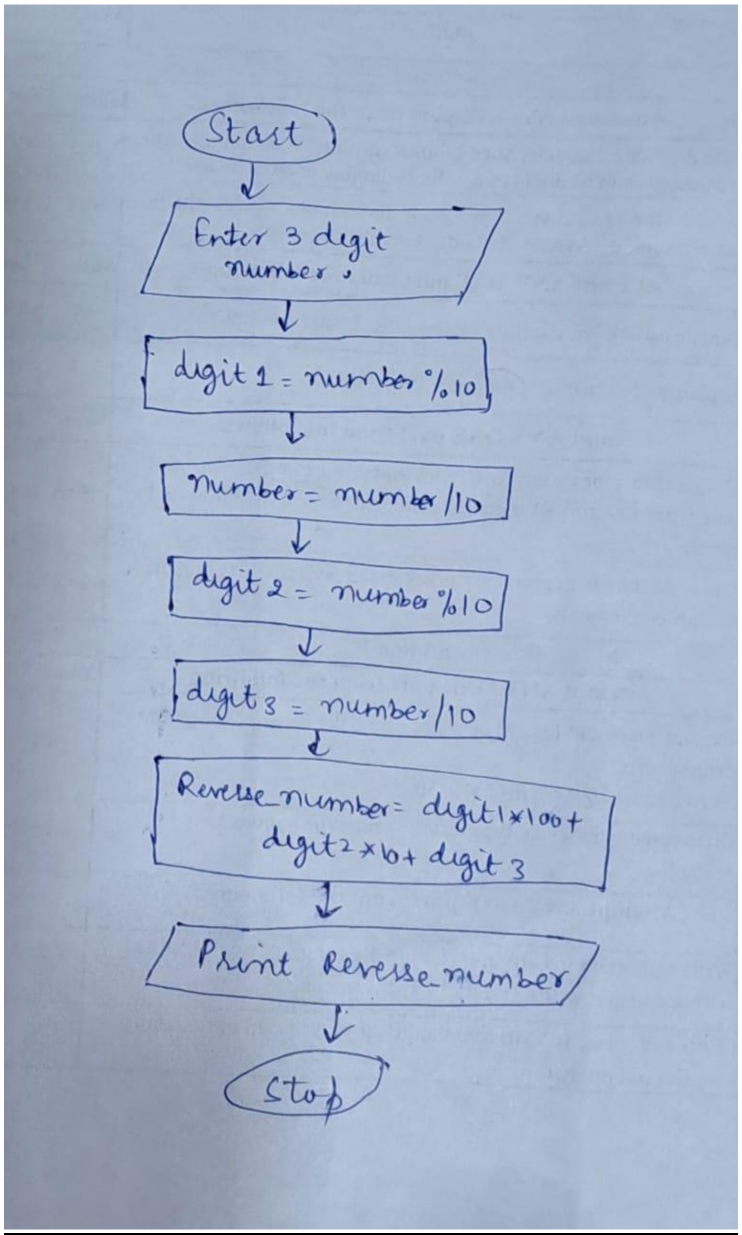
Describe the various symbols used in flow chart and draw the flow chart to find the reverse of a 3-digit number.

Flowchart: A flowchart is **visual or graphical representation of an algorithm**.



Symbol	Symbol Name	Description
	Flow Lines	Used to connect symbols
	Terminal	Used to start, pause or halt in the program logic
	Input/output	Represents the information entering or leaving the system
	Processing	Represents arithmetic and logical instructions
	Decision	Represents a decision to be made
	Connector	Used to Join different flow lines
	Sub function	used to call function

Flowchart to find the reverse of a 3 digit number



2(a)

Explain assembler, compiler, interpreter, loader and linker.

Assembler: An assembler converts programs written in assembly language into machine code. It is also referred to assembler as assembler language by some users. The source program has assembly language instructions, which is an input of the assembler. The assemble translates this source code into a code that is understandable by the computer, called object code or machine code.

.**Compiler:** The language processor that reads the complete source program written in high-level language in one go and translates it into an equivalent program in machine language is called a Compiler. Example: C, C++, C#.

In a compiler, the source code is translated to object code successfully if it is free of errors. The compiler specifies the errors at the end of the compilation with line numbers when there are any errors in the source code.

Interpreter: The programs written with the help of using one of the many high-level programming languages are directly executed by an interpreter without previously converting them to an object code or machine code, which is done line by line or statement by statement. When the interpreter is translating the source code, it displays an error message if there is an error in the statement and terminates this statement from translating process. When the interpreter removed errors on the first line, then it moves on to the next line.

Linker: A linker is special program that combines the object files, generated by compiler/assembler and other pieces of code to originate an executable file has .exe extension. In the object file, linker searches and append all libraries needed for execution of file. It regulates the memory space that will hold the code from each module. It also merges two or more separate object programs and establishes link among them.

Loader: It is special program that takes input of executable files from linker, loads it to main memory, and prepares this code for execution by computer. Loader allocates memory space to program.

2(b)

Explain structure of a C program.

Basic structure of C- Program:

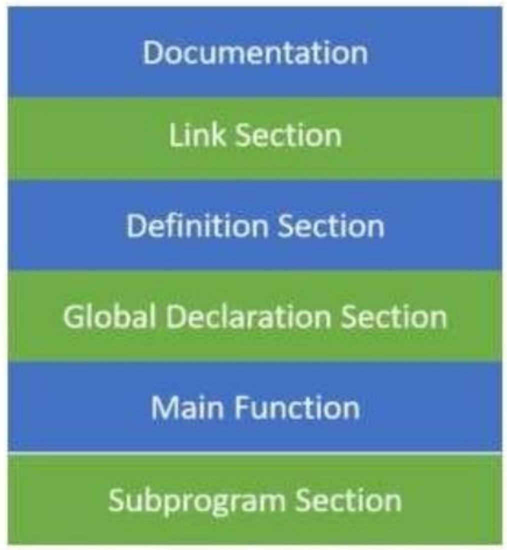


Figure: Basic Structure Of C Program

Documentation Section

- The documentation section is the part of the program where the programmer gives the details associated with the program. He usually gives the name of the program, the details of the author and other details like the time of coding and description. It gives anyone reading the code the overview of the code.

Link Section

- This part of the code is used to declare all the header files that will be used in the program.

This leads to the compiler being told to link the header files to the system libraries.

Definition Section

- In this section, we define different constants. The keyword define is used in this part.

Global Declaration Section

- This part of the code is the part where the global variables are declared. All the global

variable used are declared in this part. The user-defined functions are also declared in this part of the code.

Main Function Section

- Every C-programs needs to have the main function. Each main function contains 2 parts.

A declaration part and an Execution part. The declaration part is the part where all the variables are declared. The execution part begins with the curly brackets and ends with the curly close bracket. Both the declaration and execution part are inside the curly braces.

Sub Program Section

- All the user-defined functions are defined in this section of the program.

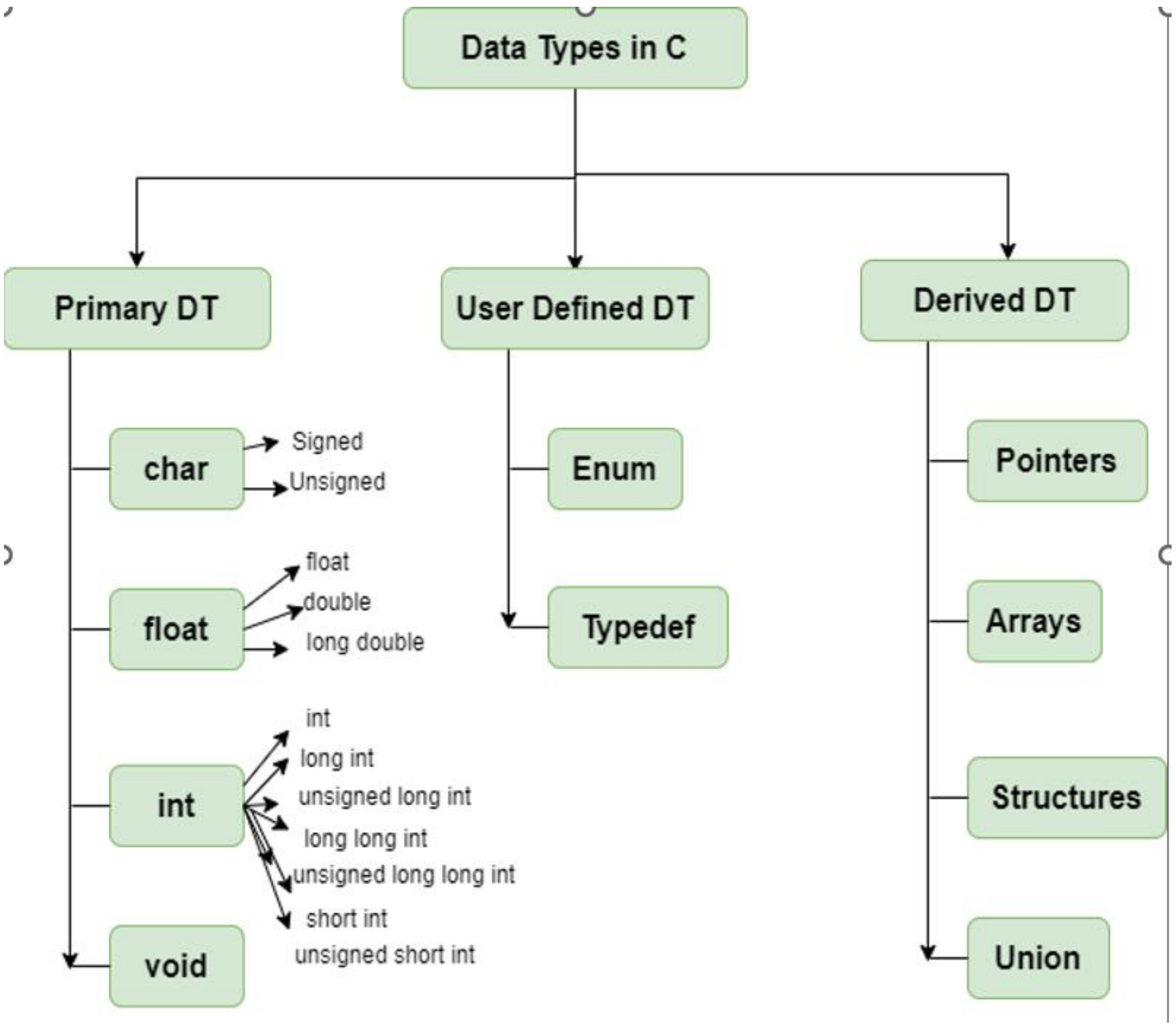
<u>**3(a)**</u>

Define data types used in C with suitable examples. Discuss data types in terms of memory occupied, format specifier and range.

<u>**Data types in C**</u>

- Each variable in C has an associated data type. It specifies the type of data that the variable can store like integer, character, floating, double, etc. Each data type requires different amounts of memory and has some specific operations which can be performed over it.
- For example,

- int a;
- Here, a is a variable of int (integer) type. The size of int is 2 bytes.



integer (int)

Integers are whole numbers that can have both zero, positive and negative values but no decimal values. For example, 0, -5, 10

We can use int for declaring an integer variable.

int age;

Here, age is a variable of type integer.

We can declare multiple variables at once . For example,

int num1,num2;

The size of int is usually 2 bytes (16 bits).

float and double

float and double are used to hold real numbers. Keyword float and double are used for declaring variables.

float sum;

double num;

char

Keyword char is used for declaring character type variables.

char a='l';

The size of the character variable is 1 byte.

void

void is an incomplete type. It means "nothing" or "no type". You can think of void as absent. For example, if a function is not returning anything, its return type should be void.

Note that, you cannot create variables of void type.

(Type specifiers) Short and long

long

If we need to use a large number, we can use a type specifier long.

long int a;

short

If we are sure, only a small integer range will be used, you can use short.

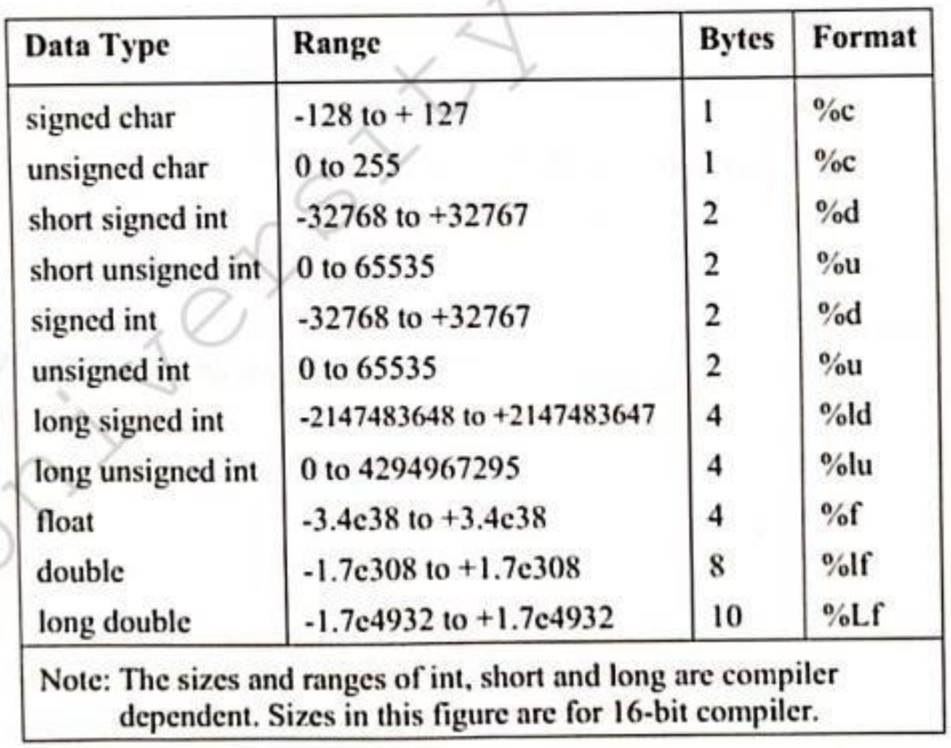
(Type Modifiers) signed and unsigned

In C, signed and unsigned are type modifiers. we can alter the data storage of a data type by using them:

signed - allows for storage of both positive and negative numbers

unsigned - allows for storage of only positive numbers

unsigned int x=15;



Data Type	Range	Bytes	Format
signed char	-128 to + 127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32768 to +32767	2	%d
short unsigned int	0 to 65535	2	%u
signed int	-32768 to +32767	2	%d
unsigned int	0 to 65535	2	%u
long signed int	-2147483648 to +2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to +3.4e38	4	%f
double	-1.7e308 to +1.7e308	8	%lf
long double	-1.7e4932 to +1.7e4932	10	%Lf

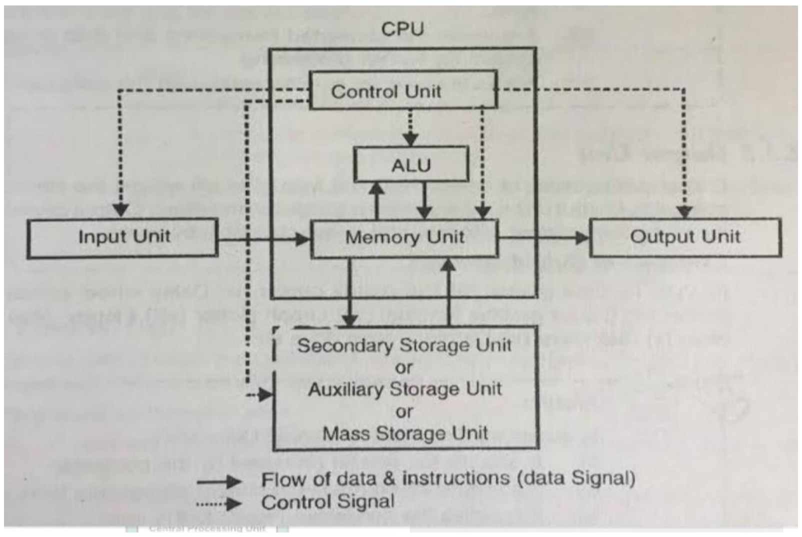
Note: The sizes and ranges of int, short and long are compiler dependent. Sizes in this figure are for 16-bit compiler.

3(b)

Draw the block diagram of a computer system. Explain its various components.

<u>Block Diagram of a digital Computer</u>

A computer is an electronic device that takes data and instructions as an input form and process it according to set of instruction and provides useful information as output.



<u>CPU (Central Procxessing Unit)</u>

CPU is the brain of Computer where all kinds of processing are done. This unit takes the input data from the input devices and process it according to set of instructions called program. The outputs of processing of the data are directed to the output devices for use in the outside world. The major function of the CPU is to store the data temporarily in the registers and perform arithmetic and logical computations. This unit also controls the operation of all other functional units of the computer system. It is just like brain that takes all major decisions, makes all sorts of calculations and directs different parts of the computer functions by activating and controlling the operations.

The Central Processing Unit is divided into three separate units for its operation. They are

a) **Arithmetic Logical Unit (ALU)** :

This unit is responsible for carrying out:

i)Arithmetical operations on data by adding subtracting, multiplying and dividing one set with another.

ii) logical operations often known by comparing by using AND, OR, NOT and exclusive OR operations which is done by analyzing and evaluating data .

iii) this unit performs increment, decrement, and shift operations as well.

b) **Control Unit (CU)**

CU acts like the supervisor. This unit is used for generating electronic signals for the synchronization of various operations. **All the related functions for program execution such as memory read, memory write, I/O read, I/O write, execution of instruction are synchronized through the control signal generated by the Control Unit.** It is responsible for coordinating various operations using time signal. The control unit determines the sequence in which computer programs and instructions are executed. It controls all the operations of the computer.

c) **Memory**

Memory is also known as storage. Its function is to store information from the human operator through the input device. The information stored may be used immediately or it can be kept for later reference for processing by CPU.

Memory is divided into two categories:

(i) Main or Primary Memory

(ii) Auxiliary or Secondary Memory

(i) **<u>Main or Primary Memory:</u>** This memory is used to store the data and program temporarily during the execution of a program. It stores program along with the data to be processed. CPU directly accesses this memory.

<u>(a)RAM(Random Access Memory):</u> . RAM is volatile memory, because it holds the data temporarily and when the power is switched off, all data stored in this memory is washed.

It is also called Read Write Memory, because it is used by CPU to temporarily load the program instruction and intermediate result. The program instructions and data keep on changing by the process of reading and writing to it.

It also called Random Access Memory. Its access time is in nano second.

(b)ROM(Read Only Memory): **ROM is non volatile memory**. It means when the power is switched off, the data or program stored in ROM is not destroyed. Whenever the power comes, the same data appears once again. This so happen because of the permanent hardware pattern used to store the data in ROM.. ROM can be used to reading or fetching of data from it. The data or programs written in ROM only once, it cannot be modified or altered.

Masked ROM

Programmable ROM

Erasable RROM

Electrically Erasable PROM

Secondary Memory: It is used to store the Operating System, Compiler, Assembler, application programs and data files. Those are not read by CPU directly. If any information is needed to processed from the secondary memory, it must be transferred to primary memory. The secondary memory is used for storage. Hard disk are used for this purpose.

Input Unit: The function of the input unit is to accept coded information from the human operator or from an electromechanical device or from other computers connected to it through the internet or by any other media.. Keyboard, Mouse, Scanners are the examples of input unit.

Output Unit: Output unit is used to represent the information processed by the digital computer. The function of the output unit is to store the processed information and display it as and when needed by the user. CRT, printers are the examples of output unit.

4(a)

Explain unary and logical operator. Find the value of following expressions:
i) 10 >>2 ii) 20 << 2 iii) 25 & 30

UNARY operator which operates on single operand

Increment (++)

The increment operator (++) is used to increment the value of the variable by 1. The increment can be done in two ways:

Preincrement:

The operator precedes the operand (e.g., ++a). The value of the operand will be changed before execution of the statement or before it is used.

Postincrement:

The operator follows the operand (e.g., a++). The value operand will be altered after execution of statement or after it is used.

Decrement (--)

The decrement operator (--) is used to decrement the value of the variable by 1. The decrement can be done in two ways:

Predecrement:

The operator precedes the operand (e.g., --a). The value of the operand will be changed before execution of the statement or before it is used.

Postdecrement:

The operator follows the operand (e.g., a--). The value operand will be altered after execution of statement or after it is used.

Logical Operators:

An expression containing logical operator return either 0 or 1 depending whether expression result true or false. This operation is commonly used in decision making programming.

&&, ||, !

Logical AND and OR is binary operator , ! Is unary operator

Assume variable A holds 10 and variable B holds 20 then:

- **&&** Called Logical AND operator. If both the operands are non zero then then condition becomes true.

(A && B) is true.

- || Called Logical OR Operator. If any of the two operands is non zero then then condition becomes true.   (A || B) is true.
- ! Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.

!(A && B) is false.

Value of 10>>2	:	**2**
Value of 20<<2	:	**80**
Value of 25&30	:	**24**

4(b)

Differentiate between typecasting and type conversion with examples.

Type cast (Explicit typecasting)	**Type Conversion (Implicit typecasting)**
It is performed explicitly by the programmer with the help of casting operator.	It is performed automatically by the compiler.
Type casting can be applied to **compatible data types** as well as **incompatible data types**.	Whereas type conversion can only be applied to **compatible datatypes**.
In type casting, casting operator is needed in order to cast a data type to another data type.	Whereas in type conversion, there is no need for a casting operator.
There is loss of data.	There is no loss of data.
It is also called as explicit typecasting.	It is also called as implicit type casting.
In typing casting, the destination data type may be smaller than the source data type, when converting the data type to another data type.	Whereas in type conversion, the destination data type can’t be smaller than source data type.
Type casting takes place during the program design by programmer.	Whereas type conversion is done at the compile time.
Type casting is also called **narrowing conversion** because in this, the destination data type may be smaller than the source data type.	Whereas type conversion is also called **widening conversion** because in this, the destination data type cannot be smaller than the source data type.

`#include<stdio.h>`
`void main()`
`{`
`float a=5.8;`
`int b;`
`b=(int)a;`
`printf("%d",b);`
`}`
 	
//Output: 5
	`#include<stdio.h>`
	`void main()`
	`{`
	`int a=5;`
	`float b;`
	`b=a;`
	`printf("%f",b);`
	`}`
	
	//Output: 5.000000

5(a)

Write a program in C to interchange the values of two variables with using and without using third variable.

//write a program in C to swap two values with using third variable

```
#include<stdio.h>

#include<conio.h>

int main()
```

```
{
int  x,y,z;
printf("enter two numbers");
scanf("%d%d",&x,&y);
z=x;
x=y;
y=z;
printf("\n after swapping value of x is \t%d\n value of y is \t%d",x,y);
return 0;
}
```

//write a program in C to swap two values without using third variable

```
#include<stdio.h>
#include<conio.h>
int main()
{
int  x,y,z;
printf("enter two numbers");
scanf("%d%d",&x,&y);
x=x+y;
y=x-y;
x=x-y;
printf("\n after swapping value of x is \t%d\n value of y is \t%d",x,y);
return 0;
}
```

5(b)

Write a program in C to find the given year is leap or not using conditional operator.

```
//find the given year is leap or not using conditional operator
#include<stdio.h>
 int main()
{
  int year;
  printf("enter year");
  scanf("%d",&year);
  (year%4==0 && year%100!=0) ? printf("leap year"): (year%400==0) ? printf("leap year") :
printf("not leap year");
}
```