Q-1 Do as directed. **[10]**

(a) In which numbering system can the binary number 1011011111000101 be easily converted to? **[01]**

a. decimal
b. hexadecimal
c. octal
d. no need to convert

(b) When a switch statement is better than multiple if statements? **[01]**

A **switch** statement is generally best to use when we have more than two conditional expressions based on a single variable of numeric type. For instance, rather than the code, the code between switch…case is easier to read and maintain. Time complexity decreases when we use switch…case statements rather than multiple if statements.

(c) Give the differences between Compiler and Interpreter. (Minimum 4 differences are required) **[02]**

	Compiler	Interpreter
1	Compiler Takes Entire program as input	Interpreter Takes Single instruction as input.
2	Intermediate Object Code is Generated	No Intermediate Object Code is Generated
3	Conditional Control Statements are Executes faster	Conditional Control Statements are Executes slower
4	Memory Requirement : More (Since Object Code is Generated)	Memory Requirement is Less
5	Program need not be compiled every time	Every time higher level program is converted into lower level program
6	Errors are displayed after entire program is checked	Errors are displayed for every instruction interpreted (if any)
7	Example : C Compiler	Example : BASIC

(d) Write a program to find the right most integer digit of a given floating point number. **[02]**

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int digit;
        float number;
        clrscr();

        printf("Enter a floating point number : ");
        scanf("%f",&number);

        digit=(int)number%10;
```

```
        printf("\nRight most digit of integer is %d",digit);
        getch();
}
```

(e) Write a program to make a simple calculator using all arithmetic operations using switch…case statement. User has to input two integer numbers and a symbol of arithmetic operator. **[04]**

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a, b;
        char d;
        clrscr();

        printf("Enter the values for a & b : ");
        scanf("%d %d",&a,&b);

        printf("\nPress a key according to your choices : ");
        printf("\nYour choices are…\n 1. '+' \n 2. '-' \n 3. '*' \n 4. '/' \n 5. '%' \n");

        d=getch();

        switch(d)
        {
                case '+':
                        printf("Sum = %d",a+b);
                        break;
                case '-':
                        printf("Subtraction = %d",a-b);
                        break;
                case '*':
                        printf("Multiplication = %d",a*b);
                        break;
                case '/':
                        printf("Division = %d",a/b);
                        break;
                case '%':
                        printf("Modulus = %d",a%b);
                        break;
        }
        getch();
}
```

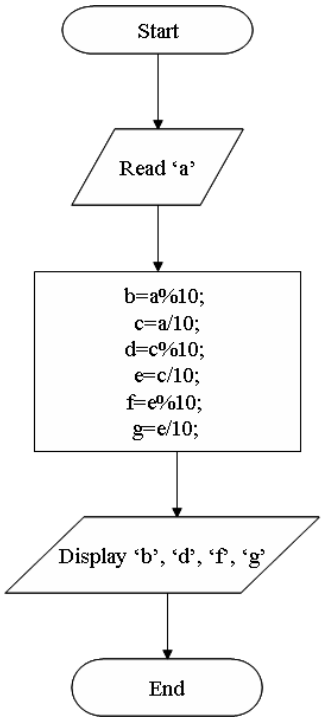
Q-2 Answer the following questions. **[10]**

(a) What do you mean by an algorithm? Draw a flow chart for finding the reverse number of any given number. Convert $(FAB)_{16}$ into octal and decimal number system. **[03]**

Algorithm: A pre-determined series of instructions fro carrying out a task in a finite number of steps.

$(FAB)_{16}$: in octal, **7653** & in decimal, **4011**

Flow chart:



OR

(a) What are different types of C tokens? Explain them in detail. **[03]**

There are **6 types of tokens in 'C'**. They are:

- **Keywords**

  Keywords are reserved words with special predefined meanings in C. These are used only for their indented purpose. They cannot be used as identifiers. Keywords are always in lowercase. Examples are auto, break, case, char, double, int.

- **Identifiers**

  Identifiers are the names by which we identify a variable, function, arrays, etc. Identifiers are case sensitive. These are user-defined names and consist of a sequence of letters and digits, with a letter as a first character. The underscore is the only symbol that can be used

in naming an identifier.

- **Constants**

  Constants are those values in C program that do not change during the execution of a program. For e.g., to calculate the percentage, one has to always multiply the marks obtained by total marks by 100. This can be set as constant as PER 100.

  There are different kinds of constants:
  Numeric and character. Numeric constants are further of Integer type and Real type while character constants are of Single Character and String type.

- **Strings**

  Character is represented as a single character enclosed within a pair of single quote. E.g., 'X', ';', '5′, 'a'. String is a sequence of characters enclosed in double quotes such as "1987″, "5 + 3″, "ANSWER".

- **Special symbols**

  These are some symbols of mathematical or functional significance in C as follows:
  {} : These opening and ending curly braces often mark the beginning and end of statement block to be processed as certain head definition of function such as main() or control statement such as if(condition).

  In expressions where operands are joined by different operators adding {} or [] braces can change the way the operands should be treated as they are highest in the operator hierarchy.

  [] : This symbol serves as Array element reference. Array cannot be defined without these braces. These braces hold the number of array elements and their reference index number. In 2-D and multi-dimensional array, these represents number of rows and columns of array elements.

  C supports some special backslash characters to format the output functions. For e.g., the symbol '\n' stands for newline character. When encountered by computer, the next instruction/output appears on next line in the C program. Other examples are '\b' for backspace, '\" for single quote, '\?' for question mark.

- **Operators**

  For processing data into useful information C variables, function references, array elements are joined together by various operators to form an expression and tells computer to perform certain mathematical or logical manipulations. The data items that operators act upon are called operands. There are 8 types of operators.

  Operators:

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and Decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators

These operators perform manipulation to expressions such as arithmetic operation of substraction, multiplication or of relational such as comparing two quantities and taking decision based on boolean value results.
E.g., +, – , <, AND, OR, ++

(b) Write a program to find sum of all integers greater than 100 less than 200 that are divisible by 7 using while loop. **[03]**

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i=101, sum=0;
        clrscr();

        printf("Numbers netween 100 and 200 which are divisible by
                7...\n");

        while(i<200)
        {
                if(i%7==0)
                {
                        printf("%d\n",i);
                        sum=sum+i;
                }
                i++;
        }
        printf("\n\n Sum = %d",sum);
        getch();
}
```

(c) Give an output of the following statements. **[04]**

a.
```
main()
{
        int a=10,b=12,x=0,c;
        c=(a>5 && b==10 || x!=1);
        printf(" \n %d",c);
  }
```
O/P: 1

b.
```
main()
{
        int x=10;
        printf("%d %d %d", --x, --x, ++x);
}
```
O/P: 9 10 11

c.
```
main()
{
        printf("%d", sizeof(double));
}
```
O/P: 8

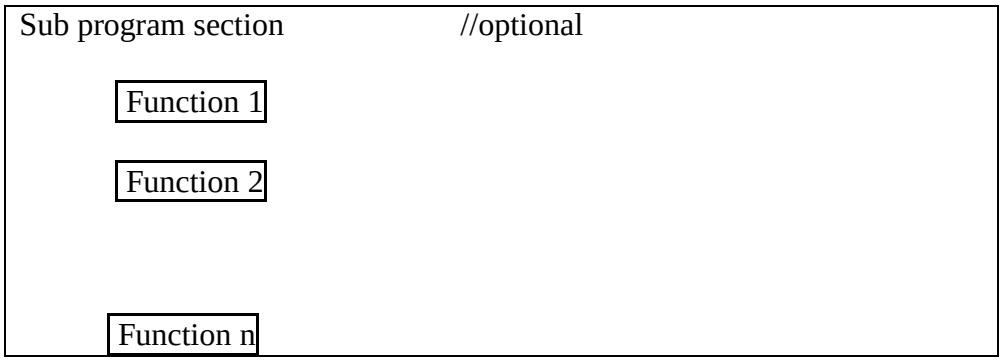
d.
```
main()
{
        int a=2, b=4, c=1, d=3, e=7, f=6, g;
        g = a + b / d * c * (f % a) – e;
        printf("%d", g);
}
```
O/P: -5

Q-3 Answer the following questions. **[10]**

(a) Explain the basic structure of C program. **[03]**

Documentation Section //optional
Link section //optional
Defining section //optional
Global declaration section //optional

Main function section //Must
{
Declaration part
Executable part.
}



- **The documentations section** consists of comment lines giving the name of the program ,the author and other details which the programmer would like to use later. These comments beginning with the two Characters * and ending with the characters*\.
- **The link section** provides to the compiler to link functions from the system library
- **The definition section** defines all symbolic constants.

There are some variables that are used in one or more functions, such variables are called global variables and are declared in the global declaration section that is outside of all the functions.

- Every C program must have one main () function section. This section can contain two parts; they are **Declaration part** and **Execution part**.

The declaration part declares all the variables used in the executable part.

There is at least one statement in the executable part.

These two parts can appear between the opening and closing braces. The program execution begins at the opening braces and ends at the closing braces. The closing brace of the function section is the logical end of the program.

All statements in the declaration and executable parts end with a semicolon.

The sub program section contains all the user defined functions that are called in the main() function. User defined functions are generally placed immediately after the main function.

A program can be viewed as a group of building blocks called functions. A function is a sub-routine that may include one or more statements designed to perform a specific task. To write a 'C' program we first create functions and then put them together. A 'C' program may contain a one or more sections as shown in figure.

(b) Give the difference between entry-controlled loop and exit-controlled loop with example. [03]

- In while loop the condition is tested first and then the statements are executed if the condition turns out to be true.
  In do while the statements are executed for the first time and then the conditions are tested,

if the condition turns out to be true then the statements are executed again.

- A do while is used for a block of code that must be executed at least once. These situations tend to be relatively rare, thus the simple while is more commonly used.
- A do while loop runs at least once even though the condition given is false while loop do not run in case the condition given is false
- In a while loop the condition is first tested and if it returns true then it goes in the loop. In a do-while loop the condition is tested at the last.
- While loop is entry control loop where as do while is exit control loop.
- **Syntax:**

<u>while loop</u>:

```
while (condition)
{
        Statements;
}
```

<u>do while loop</u>:

```
do
{
        Statements;
}while(condition);
```

- **Example:**

```
void main()
{
    int c=1, n;
    clrscr();
    scanf("%d",&n);
 /* while(c<=n)
    {
            printf("%d\n",c);
            c++;
    }*/

/*  do
    {
            printf("%d\n",c);
            c++;
    }
    while(c<=n);    */

    getch();
}
```

OR

(b) Write a program to find the factorial of a given number using while loop and input should be given by the user. [03]

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n, fact;
        fact = 1;

        printf("Enter any number : ");
        scanf("%d",&n);

        while(n>0)
        {
                fact=fact * n;
                n--;
        }

        printf("The factorial of a given number is %d", fact);
        getch();
}
```

(c) List out all the categories of operators in C. Explain any four in detail. [04]

C operators can be classified in to 6 categories:

- Arithmetic operators
- Relational operators
- Logical operators
- Assignment operators
- Increment and decrement operators
- Conditional operators
- Bitwise operator
- Special operators

- **Arithmetic operators**

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

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by denumerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0

- **Relational operators**

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

Operator	Description	Example
==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

- **Logical operators**

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

Operator	Description	Example
&&	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.

- **Assignment operators**

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

- **Increment and decrement operators**

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

Operator	Description	Example
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

- **Conditional operators**

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Syntax: exp1 ? exp2 : exp3

For e.g., x = (a > b) ? a : b;

- **Bitwise operator**

Bitwise operator works on bits and perform bit by bit operation.

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
|	Binary OR Operator copies a bit if it exists in eather operand.	(A | B) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

Assume if A = 60; and B = 13; Now in binary format they will be as follows:
A = 0011 1100
B = 0000 1101

A&B = 0000 1100
A|B = 0011 1101
A^B = 0011 0001

- **Special operators**

Operator	Description	Example
sizeof()	Returns the size of an variable.	sizeof(a), where a is integer, will return 4.
&	Returns the address of an variable.	&a; will give actaul address of the variable.
*	Pointer to a variable.	*a; will pointer to a variable.
,	Comma operator, a comma-linked list of expressions are evaluated	value = (x=10, y=5, x+y); final answer will be 15

	left to right and the value of right most expression is the value of combined expression	