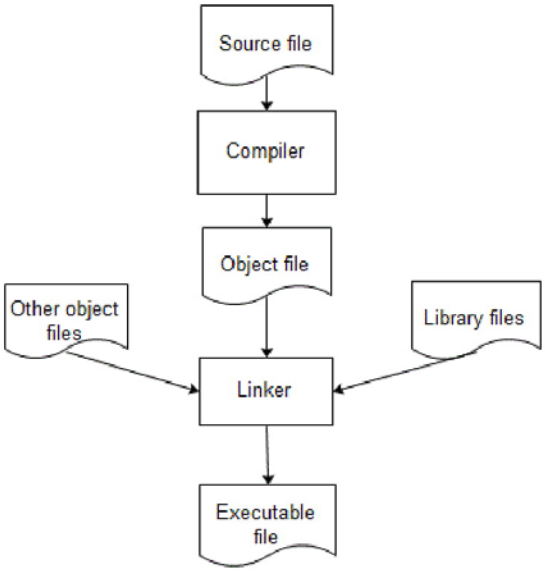
Exercise 4(A)

1.Why C is most popular language? Explain with suitable reason.

C language is very popular language because it has large numbers of features for programmer to write medium types of program. Some features of c language are given below:

a. It is a robust language with rich set of built-in functions and operators that can be used to write any complex program.
b. The C compiler combines the capabilities of an assembly language with features of a high-level language.
c. Programs Written in C are efficient and fast. This is due to its variety of data type and powerful operators.
d. It is many time faster than BASIC.
e. C is highly portable this means that programs once written can be run on another machines with little or no modification.
f. Another important feature of C program, is its ability to extend itself.
g. A C program is basically a collection of functions that are supported by C library. We can also create our own function and add it to C library.
h. C language is the most widely used language in operating systems and embedded system development today.

2.Explain compilation process with appropriate block diagram.



The steps of compilation process are given below:

Step 1: Use text editor like notepad to write your source code and save the file extension with (.c) dot c.
Step 2: Compile the program using a compiler. If compiler does not find any errors in the program, it produces an object file with .obj extension and the same name as the source code file.
Step 3: Link the program using a linker. If no errors occur, the linker produces an executable program located in a disk file with .exe extension and the same name as the object file.
Step 4: Execute the program. We should test to examine whether it functions properly.

3. Why header file files are needed in every C program? Explain.
A header file is a file that allows programmers to separate certain elements of a program's source code into reusable files. Header files commonly contain forward declarations of classes, subroutines, variables, and other identifiers. Programmers who wish to declare standardized identifiers in more than one source file can place such identifiers in a single header file, which other code can then include whenever the header contents are required. This is to keep the interface in the header separate from the implementation.
This reduces dependencies so that code that uses the header doesn't necessarily need to know all the details of the implementation and any other headers needed only for that. This will reduce compilation times and also the amount of recompilation needed when something in the implementation changes.

4. Write down the importance of header file and explain briefly about any three header files of C language.
Header files are the placeholder of commonly used functions which are frequently used during the execution of a program. Thus creating header file we are able to globally define the advantage of function available in header file. Header file are generally tag with our compiler, or these are predefine. But we can also create a header file which contains a number of functions which are commonly and frequently used in our program. Thus creating header file, we are able to use the strength of structural language and modularity.
Some header files are explained below:

a. #include<stdio.h>: it is standard input output header file. It contains the function definition of input and output functions such as scanf() and printf().
b. #include<conio.h>: It is a header file which is included in the program. This is used to use, clrscr(), getch(), etc. This header is optional header file in c program.
c. #include<math.h>: This header file is used for mathematical functions such as pow(), sqrt(), tan(), etc.

5. What is pre-processor? List any two preprocessor.
A preprocessor is a program that processes a file before it's passed down to the compiler and the linker. It permits to define some variables so that a programmer can change the program just by changing one line of code. It permits to include header files too.

Some of the common known preprocessor are C preprocessor and C++ preprocessor.

Exercise 4(B)

1. What are different character sets used in C?

Whenever we write any C program then it consists of different statements. Each C Program is set of statements and each statement is set of different c programming leximes. In C Programming each and every character is considered as single lexim.

Character set consists of:

Types	Character sets
Lowercase letters	a-z
Uppercase letters	A-Z
Digits	0-9
Special characters	!@#$%^&*
White spaces	Tab or New line or Space

2. Differentiate between identifier and keyword.

The differences are given below:

Identifier	keyword
Identifiers are names that are given to various program elements, such as variable, function and arrays.	C take some reserved word called keyword, they have predefine meaning in C.
Identifiers consist of letters and digits.	Keyword consist only letter.
Identifier's first character must be a letter.	Keyword's all character is letter.
Identifiers Upper and lowercase letter is use.	Keywords are all lowercase.
Upper and lowercase are not equivalent.	Upper and lowercase are also not equivalent.
Like: X, sum_5, _weather etc. But 4^{th} is not identifier cause identifier first character must a letter.	Like: auto, short, long etc.

3. What is primary data types used in C.

The fundamental data types are called primary data types. They are also used to build other data types. The fundamental data types are given below:

Types	Size	Range
char (or signed char)	1	-128 to 127
unsigned char	1	0 to 255
int (or signed int)	2	-32768 to 32767
unsigned int	2	0 to 65536
short int (or signed short int)	1	128 to 127
unsigned short int	1	0 to 255
long int (or signed long int)	4	-2,147,483,648 to2,147,483,647
unsigned long int	4	0 to 4,294,967,295
Float	4	3.4E - 38 to 3.4E + 38
Double	8	1.7E - 308 to 1.7E + 308
Long	10	3.4E _ 4932 to 1.1E + 4932

4.Write down the importance of variable declaration in C program.

The importance of variable declaration is to allocate memory space inside a memory of computer. Declaration does two things:

a. It tells the compiler what the variable name is.
b. It specifies what type of data the variable will hold.

5. Define the statement and also mention its types.

In computer programming a statement is the smallest standalone element of an imperative programming language which expresses some action to be carried out. A program written in such a language is formed by a sequence of one or more statements.

The types of statements of C language are given below:

a. Simple statement: A simple statement consists of an expression followed by a semicolon. The execution of a simple statement causes the expression to be evaluated. For example: a=10;
b. Compound statement: A compound statement consists of several individual statements enclosed in a pair of braces { }. It provides capability for embedding statements within other statements.

   For example:

```
{
        a=10;
```

```
        b=5;
        c=a*b;
}
```

Exercise 4(C)

1. What is operator. Define its types.

Operators are the symbols that can use for mathematical and logical operations such as +, *, -, /, %, <, <=, etc.

The types of operators are given below:

a. Arithmetic Operators
b. Relational Operators
c. Unary Operators
d. Assignment Operators
e. Logical Operators
f. Conditional Operators

2. Differentiate between binary and ternary operator.

Binary operators are operators that deal with two arguments, both generally being either variables or constants.

Operator	Use
+	addition
-	subtraction
*	multiplication
/	division
%	remainder division
=	assignment
==	Boolean equality comparison
>	Boolean greater than
<	Boolean less than
&&	Boolean AND
||	Boolean OR

Tertiary operators are operators that deal with three arguments which can be anything from a constant to a complete Boolean expression.

Syntax: (condition)? statement1: statement2

Example:
a = 10;
b = 5;
x = (a>b)? a: b;

3. Differentiate between assignment and conditional operator.

Assignment operator evaluates an expression on the right of the expression and substitutes it to the value or variable on the left of the expression.
Syntax: variable = expression
For example: a = b+c;
There are other forms of assignment operators which are listed below:

Operators	Statement	Equivalent to
+=	a+=1	a=a+1
-=	a-=1	a=a-1
*=	a*=1	a=a*1
/=	a/=1	a=a/1
%=	a%=1	a=a%1

Conditional operators are used to evaluate a condition that's applied to one or two Boolean expressions. The result of the evaluation is either true or false.
There are three conditional operators:
&& the logical AND operator. || the logical OR operator. ?: the ternary operator.
The logical AND and logical OR operators both take two operands. Each operand is a boolean expression (i.e., it evaluates to either true or false). The logical AND condition returns true if both operands are true, otherwise it returns false. The logical OR condition returns false if both operands are false, otherwise it returns true.

4. What is Special operator of C language.

The special operators are given below:

a. Comma Operator: Comma operators are used to link related expressions together.
   For example: int a, c=5,d;
b. The sizeof operator: It is a unary operator which is used in finding the size of data type, constant, arrays, structure etc.
   For example:
   int a;
   printf("Size of int=%d bytes\n", sizeof(a));

5.Write down the precedence and associativity of operators.

Table below gives the rules determining the associativity and precedence of all the operators in C. Associativity means whether an expression like x R y R z (where R is an operator such as + or <=) should be evaluated 'left-to-right' i.e. as (x R y) R z or `right-to-left' i.e. as x R (y R z). Precedence determines how an expression like x R y S z

should be evaluated (now R and S are different operators). If R has higher precedence than S, it will be evaluated as (x R y) S z, while if S has higher precedence than R it will be treated as x R (y S z).

Operators in order of precedence	Associativity
(), [], ->, .	left to right
!, ~, ++ , -, - (unary), * (indirection), & (address-of), sizeof, casts	right to left
* (multiplication), /, %	left to right
+, - (subtraction)	left to right
Â«, Â»	left to right
<, <=, > ;=, >	left to right
==, !=	left to right
& (bitwise and)	left to right
^	left to right
|	left to right
&&	left to right
||	left to right
?:	right to left
=, +=, -= etc.	right to left
,	left to right

6.Write down the type casting process.

There are two type of type conversion. They are explained below:

a. Implicit type conversion: C permits mixing of constants and variables of different types in an expression. C automatically converts any intermediate values to the proper type so that the expression can be evaluated without losing any significance. This automatic type of conversion is known as implicit type conversion.

b. Explicit type conversion: Many times there may arise a situation where we want to force a type conversion in a way that is different from automatic conversion. The process of such a local conversion is known as explicit conversion or casting a value. The general form is: (type_name) expression

7.Write down importance of library function.

There is vast importance of library functions in c programming. Some library functions are used for input/output operation, some are for string processing operations, and some are for mathematical operation and so on. Each library function can perform specific task and return specific value. This increases the understanding of program and prewritten codes are reused with a single definition of call function. Library functions are stored in respective header files. Some library functions are given below:

a. getchar(): returns the next character typed on the keyboard.
b. putchar(): outputs a single character to the screen.
c. printf(): as previously described
d. scanf(): as previously described
e. strcat(): concatenates a copy of str2 to str1
f. strcmp():compares two strings
g. strcpy(): copys contents of str2 to str1
h. isdigit(): returns non-0 if arg is digit 0 to 9
i. isalpha(): returns non-0 if arg is a letter of the alphabet
j. isalnum(): returns non-0 if arg is a letter or digit
k. islower(): returns non-0 if arg is lowercase letter
l. isupper(): returns non-0 if arg is uppercase letter

Exercise 4(D)

1.What are input/output function of C language? Explain with examples.

C programming language provides many of the built-in functions to read given input and write data on screen, printer or in any file.

a. scanf() printf() functions: scanf() can be used to get inputs from the user.
   Syntax: scanf("control string", arg1, arg2.....)
   Example: scanf("%d", &a);
   printf() can be used to display some conversion characters along with some unchanged characters. Such conversion characters may include format specifiers or escape sequences.
   Syntax: print("control string", arg1, arg2,.....)
   Example: printf("Nepal");
b. getchar() and putchar() functions: The getchar() function reads a character from the terminal and returns it as an integer.
   Syntax: variable_name = getchar()
   putchar() function prints the character passed to it on the screen and returns the same character. This function puts only single character at a time.

Syntax: putchar(variable_name);

c. gets() and puts(): The gets() function is used for completely different purpose. It reads a string from the standard input and store in the given variable. It reads a whole line of input until a newline.
   Syntax: gets(variable)
   Example: gets(name);
   The puts function is used to display text in the monitor which is stored in the variable. But variable is always string data type.
   Syntax: puts(variable)
   Example: puts(name);, puts("Nepal");

2.Differentiate between getchar() and putchar() statement.

The getchar() function reads a character from the terminal and returns it as an integer.
   Syntax: variable_name = getchar()
   Example: getchar(city);

putchar() function prints the character passed to it on the screen and returns the same character. This function puts only single character at a time.
   Syntax: putchar(variable_name);
   Example: putchar("kathmandu");, putchar(name);

3. What is gets() function? How it is different from puts() function.

The gets() function is used for completely different purpose. It reads a string from the standard input and store in the given variable. It reads a whole line of input until a newline.
   Syntax: gets(variable)
   Example: gets(name);

The puts function is used to display text in the monitor which is stored in the variable. But variable is always string data type.
   Syntax: puts(variable)
   Example: puts(name);, puts("Nepal");

Lab Works 1:

1. WAP to input any number and display its square root.

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
{
int x,y;
printf("Enter a number:");
scanf("%d",&x);
y = sqrt(x);
printf("The Square root is: %d", y);
getch();
}
```

Output:

Enter a number:100

The Square root is: 10

2. **WAP to input length and find its cube where volume $v=l^3$.**

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
{
int l,v;
printf("Enter the length of cube:");
scanf("%d",&l);
v = pow(l,3);
printf("The volume of cube is: %d", v);
getch();
}
```

Output:

Enter the length of cube: 5

The volume of cube is: 125

3. WAP to input length and breadth and calculate area and perimeter of rectangle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int l,b,a,p;
printf("Enter the length and breadth of rectangle: ");
scanf("%d %d",&l,&b);
a = l * b;
p = 2*(l+b);
clrscr();
printf("The area of rectangle is: %d", a);
printf("\nThe perimeter of rectangle is: %d", p);
getch();
}
```

Output:

Enter the length and breadth of rectangle: 15

10

The area of rectangle is: 150

The perimeter of rectangle is: 50

4.WAP to input u,t and a calculate distance using $s=ut+1/2at^2$.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float u,t,a,s;
printf("Enter the values of 'u', 't' and 'a' respectively:\n");
scanf("%f %f %f",&u,&t,&a);
s = (u*t)+(a*t*t)/2;
printf("The distance is : %f", s);
getch();
}
```

Output:

Enter the values of 'u', 't' and 'a' respectively:

5

4

3

The distance is : 44.000000

5. WAP to input radius and calculate area and circumference of circle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float r,a,c;
printf("Enter the values of radius:\n");
scanf("%f",&r);
a = 3.14*r*r;
c = 2*3.14*r;
printf("The area of cicle is : %f", a);
printf("\nThe circumference of cicle is : %f", c);
getch();
}
```

6. WAP to input temperature in C^0 and convert in to Fahrenheit. [Hint: f =1.8c+32].

```
#include<stdio.h>
#include<conio.h>
void main()
{
float c,f;
printf("Enter the temperature in Centigrade:\n");
scanf("%f",&c);
f = 1.8*c+32;
printf("The temperature in Fahrenheit : %f", f);
getch();
}
```

Output:

Enter the temperature in Centigrade:

74

The temperature in Fahrenheit : 165.199997

7.WAP to calculate sum of two distances and the distance is measured in term of feet and inch.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float a1,b1,a2,b2,a3,b3;
printf("Enter the distance of first measurement in feet and inch respectively:\n");
scanf("%f %f",&a1,&b1);
printf("\nEnter the distance of second measurement in feet and inch respectively:\n");
scanf("%f %f",&a2,&b2);
a3 = a1 + a2;
b3 = b1 + b2;
clrscr();
printf("The resultant measurement is : %f feet and %f inch", a3,b3);
getch();
}
```

8.WAP to enter number of days and convert into years, months and days.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int day,year,month;
printf("Enter the number of days:\n");
```

```
scanf("%d",&day);
year = day/365;
day = day%365;
month = day/30;
day = day%30;
clrscr();
printf("The result is %d years, %d months and %d days", year,month,day);
getch();
}
```

9.Find the output of given program.

Program-1:

```
#include<stdio.h>
#include<conio.h>
void main()
{
float x;
int x1 = 7;
int x2 = 2;
x = (float)x1/x2;
printf("%f",x);
getch();
}
```

Output

3.500000

Program-2:

```
#include<stdio.h>
#include<conio.h>
void main()
{
float x;
int x1 = 7;
int x2 = 2;
x = (float)(x1/x2);
printf("%f",x);
getch();
}
```

Output

3.000000

10. **Find the output of given program.**

Program-1:

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

```
#include<conio.h>
void main()
{
int j;
int i = 5;
j = ++i;
        printf("i=%d\nj=%d"Â□,i,j);
        getch();
}
```

Output
i=6
j=6

Program-2:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int j;
int i = 5;
j = i++;
        printf("i=%d\nj=%d",i,j);
getch();
}
```

Output
i=6
j=5

Exercise 4(E)

1.What do you mean by control structures? Discuss about its application in programming.

Control structures are those programming constructs which control the flow of program statements execution.

Applications of control structures are given below:

a. Execute program statements based on the given condition: it controls the flow of program statement execution based on the condition check. It can be used in different forms as below:

i. if statement
ii. if else statement
iii. if else if statement

iv. nested if else statement

b. Execute program statements, repeatedly: This control structure is used where the block of statements or single statement is needed to execute repeatedly for specified number of times or till the given condition is satisfied. Its forms are:

i. while loop
ii. do while loop
iii. for loop

c. Choose an option and perform the task accordingly: There may be need to perform an operation when specific fixed value condition is matched. In this case, there is limited number of options to be performed. This operation is carried by switch control statement.

2. Explain different types of control structures with syntax and flowchart.

The different types of control structures are described below:

a. Selective control structure: In selective control structure, selection is made on the basis of condition. We have options to go when the given condition is true or false. It can be categorized in two types:

i. Conditional statements: It is the most common decision making control structure which controls the flow of program statement execution based on the condition check. It can be used in following forms:

- if statement
- if else statement
- if else if statement
- nested if else statement

ii. Switch-case statements: This statement is a multipath decision making statement that allows selection and execution of a particular block of statements from several blocks of statements based upon the value of expression which is included within switch statement and branches accordingly.

b. Looping control structure: Looping is the process of executing the same program statement or block of program statements repeatedly for specified number of times or till the given condition is satisfied. It has following types:

i. while loop
ii. do while loop
iii. for loop

c. Jumping statements: Jumping statements are particularly used to jump execution of program statements from one place to another place inside a program. Following are the jumping statements defined in C programming:

i. break
ii. continue
iii. goto

3. Discuss nested loop, infinite loop and loop interruption statement with statement with example.

C programming language allows using one loop inside another loop called nested loop.

Syntax:

```
do
{
statement(s);
do
{
statement(s);
}while( condition );

}while( condition );
```

Example:

```
#include <stdio.h>
int main ()
{
        int i, j;
        for(i=2; i<100; i++) {
                for(j=2; j <= (i/j); j++)
                        if(!(i%j)) break; // if factor found, not prime
                                if(j > (i/j)) printf("%d is prime\n", i);
        }
        return 0;
}
```

We can use infinite loop in C to iterate loop for infinite times. We can even use infinite loop for operation in which we cannot decide how many iteration does it take to compile time.

Example:

```
int main()
{
        int i = 0;
        while (1)
        {
                printf("Nepal");
                i++;
        }
}
```

Loop interruption statements are also known as jumping statements. These are used to jump execution of program statements from one place to another place or out of the looping statements. Loop interruption statements are break, continue and goto.

Example:

```
void main()
{
        int i;
        for(i=1;i<=10;i++)
        {
                If(i==5)
```

```
                {
                        continue;
                }
                else
                {
                        Printf("Nepal");
                }
        }
}
```

4.Differentiate between while loop and do while loop.

The "for loop" loops from one number to another number and increases by a specified value each time.

The "for loop" uses the following structure:

```
for (Start value; continue or end condition; increase value)
        statement
```

Example:

```
#include<stdio.h>
int main()
{
        int i;
        for (i = 0; i < 10; i++)
        {
                printf ("Hello\n");
                printf ("World\n");
        }
        return 0;
}
```

The while loop can be used if you donâ€™t know how many times a loop must run. Here is an example:

```
#include<stdio.h>
int main()
{
int counter, howmuch;

scanf("%d", &howmuch);
counter = 0;
while ( counter < howmuch)
{
counter++;
printf("%d\n", counter);
}
return 0;
}
```

5. Difference between while loop and do while loop.

while loop	do while loop
Condition is checked in beginning	Condition is checked at the end
It is not terminated with semicolon	It is terminated with semicolon
Statements are not executed when the condition is false	Statements are executed once even the condition is false
It uses keyword â€˜whileâ€™	It uses keyword â€˜doâ€™ and â€˜whileâ€™

Syntax:
while
{
statements;
}
	Syntax:
	do
	{
	statements;
	}while(condition);

6. Differentiate between break and continue statement.

C break statement terminates any type of loop e.g., while loop, do while loop or for loop. The break statement terminates the loop body immediately and passes control to the next statement after the loop. The break statement is only meaningful when you put it inside a loop body, and also in the switch case statement. We often use the break statement with the if statement, which specifies the condition to terminate the loop.

Example:

```
#include <stdio.h>
int main()
{
        char key;
        printf("Press any key or E to exit:\n");
        while(1)
{
                scanf("%c", &key);
                if (key == 'E' || key == 'e')
                        break;
        }
        printf("Goodbye!\n");
}
```

C continue statement skips the rest of the current iteration in a loop and returns to the top of the loop. The continue statement works like a shortcut to the end of the loop body.

Example:

```
void main()
{
        int i;
```

```
        for(i=1;i<10;i++)
        {
                If(i==5)
                {
                        continue;
                }
                printf("Nepal");
        }
}
```

Lab Works 2:

1.WAP To find area of circle and circumference[hint: real circle is a circle having positive radius].

```
#include<stdio.h>
#include<conio.h>
void main()
{
float r,a,c;
printf("Enter the radius:\n");
scanf("%f",&r);
if(r <= 0)
{
printf("Number must be positive.\n");
}
else
{
a = 3.14*r*r;
c = 2*3.14*r;
printf("The area of real circle is: %f\n", a);
printf("The circumference of real circle is: %f\n", c);
}
getch();
}
```

2. WAP to input any number and check whether given number is even or odd.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int x;
printf("Enter any number:\n");
scanf("%d",&x);
if(x%2==0)
{
printf("Number is Even\n");
```

```
}
else
{
printf("Number is Odd\n");
}
getch();
}
```

3.WAP to read a number and check whether it is positive, negative or zero.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float x;
printf("Enter any number:\n");
scanf("%f",&x);
if(x > 0)
{
printf("Number is Positive\n");
}
else if (x < 0)
{
printf("Number is Negative\n");
}
else
{
printf("Number is Zero\n");
}
getch();
}
```

4. WAP to input cp and sp and find loss, profit or Neither Loss nor Profit.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float cp,sp,x;
printf("Enter costprice:\n");
scanf("%f",&cp);
printf("Enter sellingprice:\n");
scanf("%f",&sp);
x = cp - sp;
if(x > 0)
{
printf("There is Loss\n");
}
```

```
else if (x < 0)
{
printf("There is Profit\n");
}
else
{
printf("Neither Loss nor Profit\n");
}
getch();
}
```

5. WAP to input any 3 numbers and find greatest and smallest.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float x,y,z,largest,smallest;
printf("Enter any 3 numbers:\n");
scanf("%f %f %f",&x,&y,&z);
if(x>=y && x>=z)
{
largest = x;
}
else if (y>=x && y>=z)
{
largest = y;
}
else
{
largest = z;
}
if(x<=y && x<=z)
{
smallest = x;
}
else if (y<=x && y<=z)
{
smallest = y;
}
else
{
smallest = z;
}
printf("The largest number is : %f\n",largest);
printf("The smallest number is : %f\n",smallest);
getch();
```

```
}
```

6. **WAP to input any 3 numbers and find middle number.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
float x,y,z,middle;
printf("Enter any 3 numbers:\n");
scanf("%f %f %f",&x,&y,&z);
if((x>=y && x<=z) || (x<=y && x>=z))
{
middle = x;
}
else if ((y>=x && y<=z) || (y<=x && y>=z))
{
middle = y;
}
else
{
middle = z;
}
printf("The middle number is : %f\n",middle);
getch();
}
```

7.WAP to calculate and display real root of a quadratic equation.[Hint: check the condition(b^2-4ac)>0 for real root]

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
{
float a,b,c,x1,x2;
printf("Enter the values of a,b and c respectively:\n");
scanf("%f %f %f",&a,&b,&c);
if((b*b-4*a*c)<0)
{
printf("Roots are not Real\n");
}
else
{
x1 = ((-b)+sqrt((b*b-4*a*c)))/(2*a);
x2 = ((-b)-sqrt((b*b-4*a*c)))/(2*a);
printf("The 1st root is: %f\n",x1);
```

```
printf("The 2nd root is: %f\n",x2);
}
getch();
}
```

8.WAP to find the commission amount on the basis of sales amount as per the following condition.

Sales amount	**Commision**
0-1000	5%
1001-2000	10%
>2000	12%

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
{
float sales,commission;
printf("Enter the Sales amount:\n");
scanf("%f",&sales);
if(sales<=1000)
{
commission = sales * 0.05;
}
else if (sales<=2000)
{
commission = sales * 0.1;
}
else
{
commission = sales * 0.12;
}
printf("Commission amount is: %f\n",commission);
getch();
}
```

9.Write interactive program the talks length and bredth and performs the following task

a. Area of rectangle **b.** Perimeter of rectangle
c. Exit

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
```

```
int main()
{
float l,b,a,p;
int choice;
printf("Enter Length and Breadth of rectangle respectively:\n");
scanf("%f %f",&l,&b);
printf("Enter '1' to calculate Area of reactangle:\n");
printf("Enter '2' to calculate Perimeter of reactangle:\n");
printf("Enter any other key to Exit:\n");
scanf("%d",&choice);
if(choice == 1)
{
a = l * b;
printf("Area of rectangle is: %f",a);
}
else if(choice == 2)
{
p = 2 * (l + b);
printf("Perimeter of rectangle is: %f",p);
}
else
{
return(0);
}
}
```

10. WAP that takes a number less than 10 and display its multiplication table.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int num;
printf("Enter any number less than 10:\n");
scanf("%d",&num);
if(num>=10)
{
printf("The number must be less than 10\n");
}
else
{
printf("The multiplication table of %d is \n:",num);
printf("----------------------------------------\n ");
for(int i=1;i<=10;i++)
{
int value = num * i;
printf("%d x %d = %d\n",num,i,value);
```

```
}
}
getch();
}
```

Lab Works 3: page-158

1.WAP to calculate and display the following series 1 3 5 …….to 10^{th} term.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int x = 1;
for(int i=1;i<=10;i++)
{
printf("%d\t",x);
x=x+2;
}
getch();
}
```

Output:

1 3 5 7 9 11 13 15 17 19

2. WAP to diaplsy square series of first 'n' natural numbers.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int n;
printf("Enter the value of 'n':\n");
scanf("%d",&n);
printf("The series is:\n");
printf("----------------\n");
for(int i=1;i<=n;i++)
{
printf("%d\n",(i*i));
}
getch();
}
```

Output:

```
nter the value of 'n':  5
The series is:
----------------
```

1
4
9
16
25

3.WAP to calculate factorial of any number given by user.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int c,n,fact = 1;
printf("Enter a number to calculate it's factorial\n");
scanf("%d",&n);
for (c=1;c<=n;c++)
{
fact = fact * c;
}
printf("Factorial of %d = %d\n", n, fact);
getch();
}
```

Output:

```
Enter a number to calculate it's factorial   3
Factorial of 3 = 6
```

4.WAP to calculate and display the multiplication table of a number.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int num;
printf("Enter any number less than 10:\n");
scanf("%d",&num);
printf("The multiplication table of %d is:\n",num);
printf("----------------------------------------\n" );
for(int i=1;i<=10;i++)
{
int value = num * i;
printf("%d x %d = %d\n",num,i,value);
}
getch();
}
```

Output:

```
Enter any number less than 10:    5
The multiplication table of 5 is:
----------------------------------------
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

5.WAP to display Fibonacci series having 'n' terms.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=0,b=1,c=1,n;
printf("Enter the value of n:\n");
scanf("%d",&n);
for (int i=1;i<=n;i++)
{
printf("%d\t",c);
c = a + b;
a = b;
b = c;
}
getch();
}
```

Output:

```
Enter the value of n: 5
1       1       2       3       5
```

6. WAP to check the given number is palindrome or not.

```
#include<stdio.h>
#include<conio.h>
void main()
```

```
{
int n,temp,sum=0,d;
printf("Enter any number:\n");
scanf("%d",&n);
temp = n;
while(n!=0)
{
d = n%10;
sum = sum*10+d;
n = n/10;
}
if(temp == sum)
{
printf("The number is Palindrome\n");
}
else
{
printf("The number is not Palindrome.\n");
}
getch();
}
```

Output:
Enter any number: 121
The number is Palindrome

7. WAP to calculate sum of the following:
Sum= 1+1/2+1/3+1/4….+1/n

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n;
float sum=0;
printf("Enter the number of term 'n':\n");
scanf("%d",&n);
for(int i=1;i<=n;i++)
{
sum = sum + (float)1/i;
}
printf("The sum value is: %f",sum);
getch();
}
```

Output:
Enter the number of term 'n': 5
The sum value is: 2.283334

8.WAP to display the following output.

1 2 3 4 5
2 4 6 8 10
3 6 9 12 15

```
#include<stdio.h>
#include<conio.h>
void main()
{
for(int i=1;i<=3;i++)
{
for(int j=i;j<=i*5;j=j+i)
{
printf("%d\t",j);
}
printf("\n");
}
getch();
}
```

9.WAP to check the whether the given number is prime or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,c=2;
printf("Enter a number to check if it is prime\n");
scanf("%d",&n);
for(c=2;c<=n-1;c++)
{
if(n%c==0)
{
printf("%d is not prime.\n", n);
break;
}
}
if(c==n)
printf("%d is prime.\n", n);
getch();
```

```
}
```

Output:
Enter a number to check if it is prime 8
8 is not prime.

10. WAP to display the prime series 2 3 5 7 11 up to n .

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i=3,count,c;
printf("Enter the number of prime numbers required\n");
scanf("%d",&n);
if(n>=1)
{
printf("First %d prime numbers are :\n",n);
printf("2\n");
}
for(count=2;count<=n;)
{
for(c=2;c<=i-1;c++)
{
if(i%c==0)
break;
}
if(c==i)
{
printf("%d\n",i);
count++;
}
i++;
}
getch();
}
```

Exercise 4(F)

1.What is array. Write difference between one dimension array and Two dimensional array.

An array is a collection of similar type of data items under the common name. An array is a collective name given to a group of similar quantities.

2.

One dimensional array	Two dimensional array
It consists of only one subscript	It consists of two subscripts
Maximum size will be sized of array which we define in program	Maximum size will be the product of row and column size of array which we define in program
It stores data either row wise or column wise	It stores data in matrix form that is row wise and column wise
Syntax: data_type array_name[array_size];	Syntax: data_type array_name[row_size][column_size];
Example: char name[10];	Example: char name[10][15];

3.Write down Importance of an array.

The importances of an array are as follows:

a. Handling similar types of data in a program.
b. Solving problems like sorting, indexing, etc.
c. Solving matrix related problems.
d. Manipulation of graphic which is an array of pixels.

4.What is string in c. Define string handling function.

String in C is the set or collection of characters, digits and symbol enclosed in quotation marks.

Strings handling functions are defined below:

a. strlen() function: this function returns the length of a string which takes the string name as an argument.
   Example:
   char name[10] = "Nepal";
   int len = strlen(name);
b. strrev() function: this function is used to reverse the given string.
   Example:
   char name[10] = "Nepal";
   strrev(name);
c. strcat() function: this function is used to join one string into other string.
   Example:
   char name1[5] = "abc";
   char name2[5] = "xyz";
   strcat(name1, name2);
d. strlwr() function: this function is used to convert upper case characters to string into lower case characters.
   Example:
   char name = "Nepal";
   strlwr(name);

5.
The action of the fundamental operations on strings, including their creation, concatenation, the extraction of string segments, string matching, their comparison, discovering their length, replacing substrings by other strings, storage, and input/output is called string manipulation.
strcpy copies one string into another. It takes the form :
strcpy(string1,string2);
It assigns the contents of string2 to string1. string2 may be a character array variable or a string constant.
The strcat() function joins two strings together. It takes the following form :
strcat(string1,string2);
Where string1 & string two are character arrays.
When the strcat function is executed, string two is appended to string1. It does so by removing the null character at the end of string1 and placing string2 from there. The string at string2 remains unchanged. Size of string1 should be enough to accommodate the final string. strcat function may also append a string constant to a string variable.

Lab Works 4:

1.WAP to input 5 subject marks and find average using array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float a[5],avg=0,total=0;
printf("Enter the marks of 5 subjects:\n");
for(int i=0;i<5;i++)
{
scanf("%f",&a[i]) ;
total = total + a[i];
}
avg = total/5;
printf("The total marks is: %f\n",total);
printf("The average marks is: %f\n",avg);
getch();
}
```

Output:

```
Enter the marks of 5 subjects:
50
60
70
80
90
```

The total marks is: 350.000000
The average marks is: 70.000000

2.WAP to input age of 100 person and display age between 50 and 60.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],count=0;
printf("Enter the ages of 100 persons:\n");
for(int i=0;i<5;i++)
{
scanf("%d",&a[i]) ;
if(a[i]>50 && a[i]<60)
{
count++;
}
}
printf("The number of persons in age group 50-60 years are: %d\n",count);
getch();
}
```

Output:

Enter the ages of 100 persons:
55
62
57
58
63
The number of persons in age group 50-60 years are: 3

3.WAP to input n numbers and display the sum of even and odd number respectively.

```
#include<stdio.h>
#include<conio.h>
void main()
```

```
{
int n,evensum=0,oddsum=0;
printf("Enter the value of n:\n");
scanf("%d",&n);
int a[100];
printf("Enter the numbers:\n");
for(int i=0;i<n;i++)
{
scanf("%d",&a[i]) ;
if(a[i]%2==0)
{
evensum = evensum + a[i];
}
else
{
oddsum = oddsum + a[i];
}
}
printf("The sum of even numbers are: %d\n",evensum);
printf("The sum of odd numbers are: %d\n",oddsum);
getch();
}
```

4.WAP to input n number and find greatest ,smallest and average number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n;
float x,min,max,avg;
printf("How many numbers(n) you going to enter:\n");
scanf("%d",&n);
printf("Enter the numbers:\n");
scanf("%f",&x);
max=x;
min=x;
for(int i=2;i<=n;i++)
{
scanf("%f",&x);
if(x>max)
{
max = x;
}
```

```
else if(x<min)
{
min = x;
}
}
avg = (min+max)/2;
printf("The Largest Number is %f",max);
printf("The Smallest Number is %f",min);
printf("The Average Number is %f",avg);
getch();
}
```

5.WAP to take salary of 100 employee and print the salary of the employee in ascending order.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i, j, a, number[100];
printf("Enter the salary of 100 employees:\n");
for (i = 0; i < 100; ++i)
scanf("%d", &number[i]);
for (i = 0; i < 100; ++i)
{
for (j = i + 1; j < 100; ++j)
{
if (number[i] > number[j])
{
a = number[i];
number[i] = number[j];
number[j] = a;
}
}
}
printf("The salaries arranged in ascending order are given below:\n");
for (i = 0; i < 100; ++i)
printf("%d\n", number[i]);
getch();
}
```

6.WAP to transpose the max matrix.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int m,n,c,d,matrix[10][10],transpose[10][10];

printf("Enter the number of rows and columns of matrix:\n");
scanf("%d%d",&m,&n);
printf("Enter the elements of matrix: \n");
for( c = 0 ; c < m ; c++ )
{
for( d = 0 ; d < n ; d++ )
{
scanf("%d",&matrix [c][d]);
}
}
for( c = 0 ; c < m ; c++ )
{
for( d = 0 ; d < n ; d++ )
{
transpose[d][c] = matrix[c][d];
}
}
printf("Transpose of entered matrix :-\n");
for( c = 0 ; c < n ; c++ )
{
for( d = 0 ; d < m ; d++ )
{
printf("%d\t",transpose [c][d]);
}
printf("\n");
}
getch();
}
```

Output:

```
Enter the number of rows and columns of matrix:
2
2
Enter the elements of matrix
5
2
4
```

3
Transpose of entered matrix :-
5 4
2 3

7. WAP to calculate sum of diagonal matrix.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10][10],i,j,sum=0,m,n;
printf("Enter the row and column of matrix:\n");
scanf("%d %d",&m,&n);
printf("Enter the elements of matrix:\n");
for(i=0;i<m;i++)
for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
printf("The matrix is:\n");
for(i=0;i<m;i++)
{
printf("\n");
for(j=0;j<m;j++)
{
printf("%d\t",a[i][j] );
}
}
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
if(i==j)
sum=sum+a[i][j];
}
}
printf("\n\nSum of the diagonal elements of a matrix is: %d",sum);
getch();
}
```

Output:
Enter the row and column of matrix:
3
3
Enter the elements of matrix:
4

5
6
8
7
9
5
4
5
The matrix is:

4 5 6
8 7 9
5 4 5

Sum of the diagonal elements of a matrix is: 16

8.WAPWAP to add two matrices supplying by elements by user.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int m, n, c, d, first[10][10], second[10][10], sum[10][10];
printf("Enter the number of rows and columns of matrix\n");
scanf("%d%d", &m, &n);
printf("Enter the elements of first matrix\n");
for ( c = 0 ; c < m ; c++ )
for ( d = 0 ; d < n ; d++ )
scanf("%d", &first[c][d]);
printf("Enter the elements of second matrix\n");
for ( c = 0 ; c < m ; c++ )
for ( d = 0 ; d < n ; d++ )
scanf("%d", &second[c][d]);
for ( c = 0 ; c < m ; c++ )
for ( d = 0 ; d < n ; d++ )
sum[c][d] = first[c][d] + second[c][d];
printf("Sum of entered matrices:-\n");
for ( c = 0 ; c < m ; c++ )
{
for ( d = 0 ; d < n ; d++ )
printf("%d\t", sum[c][d]);
printf("\n");< /P>
}
getch();
```

```
}
```

Lab Works 5: String

1.WAP to find the length of string without built in function.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char a[100];
int length;
printf("Enter a string to calculate it's length\n");
gets(a);
for(int i =0;a[i]!='\0';++i)
length = i+1;
printf("Length of entered string is = %d\n",length);
getch();
}
```

2.WAP to reverse the given string without using built in function.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char str[100], temp;
int i, j = 0;
printf("Enter the string:\n");
gets(str);

i = 0;
j = strlen(str) - 1;

while (i < j) {
temp = str[i];
str[i] = str[j];
str[j] = temp;
i++;
j--;
}

printf("Reverse string is :%s", str);
getch();
}
```

3. WAP to concatenate two given string without using built in function.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char str1[25],str2[25];
int i=0,j=0;
printf("\nEnter First String:");
gets(str1);
printf("\nEnter Second String:");
gets(str2);
while(str1[i]!='\0')
i++;
while(str2[j]!='\0')
{
str1[i]=str2[j];
j++;
i++;
}
str1[i]='\0';
printf("\nConcatenated String is %s",str1);
getch();
}
```

4. WAP for the following problems using built in function.
Strlen(),strrev(), strcmp(), strcpy(), and strcat(),

a. strlen(): length of the string is calculated

```
#include <stdio.h>
#include <string.h>
int main( )
{
int len;
char array[20]="khullakitab.com" ;
len = strlen(array) ;
printf ( "\string length = %d \n" , len ) ;
return 0;
}
```

b. strrev(): reversing input string

```
#include<stdio.h>
#include<string.h>
int main()
{
char name[30] = "Hello";
printf("String before strrev( ) : %s\n",name);
printf("String after strrev( ) : %s",strrev(name));
```

```
return 0;
}
```

c. strcmp(): comparing two string

```
#include <stdio.h>
#include <string.h>
int main ()
{
char str1[15];
char str2[15];
int ret;
strcpy(str1, "abcdef");
strcpy(str2, "ABCDEF");
ret = strcmp(str1, str2);
if(ret < 0)
{
printf("str1 is less than str2");
}
else if(ret > 0)
{
printf("str2 is less than str1");
}
else
{
printf("str1 is equal to str2");
}
return(0);
}
```

d. strcpy(): copying one string to another string

```
#include <stdio.h>
#include <string.h>
int main()
{
char src[40];
char dest[100];
memset(dest, '\0', sizeof(dest));
strcpy(src, "This is tutorialspoint.com");
strcpy(dest, src);
printf("Final copied string : %s\n", dest);
return(0);
}
```

e. strcat(): concatenating two string inputs

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main(void)
{
```

```
char str1[25],str2[25];
printf("\nEnter First String:");
gets(str1);
printf("\nEnter Second String:");
gets(str2);
strcat(str1,str2);
printf("\nConcatenated String is %s",str1);
getch();
}
```

5. WAPto convert the given line of text into upper case.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char text[100];
printf("Enter the string:\n");
gets(text);
printf("The string in uppercase is: %s\n",strupr(text));
getch();
}
```

6.WAP to input line of text and count the number of digit, vowels , consonants, white space and other characters.

```
#include<stdio.h>
int main()
{
char line[150];
int i,v,c,ch,d,s,o;
o=v=c=ch=d=s=0;
printf("Enter a line of string:\n");
gets(line);
printf("\n");
for(i=0;line[i]!='\0';++i)
{
if(line[i]=='a' || line[i]=='e' || line[i]=='i' || line[i]=='o' || line[i]=='u' || line[i]=='A' || line[i]=='E'
|| line[i]=='I' || line[i]=='O' || line[i]=='U')
++v;
else if((line[i]>='a'&& line[i]<='z') || (line[i]>='A'&& line[i]<='Z'))
++c;
else if(line[i]>='0'&&c<='9')
++d;
else if (line[i]==' ')
++s;
}
```

```
printf("Vowels: %d",v);
printf("\nConsonants: %d",c);
printf("\nDigits: %d",d);
printf("\nWhite spaces: %d",s);
return 0;
}
```

7. WAP to swap two two given strings.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char first[100], second[100],temp[100];
printf("Enter the first string\n");
gets(first);
printf("Enter the second string\n");
gets(second);
printf("\nBefore Swapping\n");
printf("First string: %s\n",first);
printf("Second string: %s\n\n",second);
strcpy(temp,first);
strcpy(first,second);
strcpy(second,temp);
printf("After Swapping\n");
printf("First string: %s\n",first);
printf("Second string: %s\n",second);
getch();
}
```

8. WAP to check whether the given string is palindrome or not.

```
#include <stdio.h>
#include <string.h>
int main()
{
char a[100], b[100];
printf("Enter the string to check if it is a palindrome\n");
gets(a);

strcpy(b,a);
strrev(b);
if( strcmp(a,b) == 0 )
printf("Entered string is a palindrome.\n");
else
printf("Entered string is not a palindrome.\n");
return 0;
```

```
}
```

9.WAP to input the names of 10 students and sort them alphabetical order.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char name[10][25], tname[10][25], temp[25];
int i, j;
printf("Enter name of 10 students:\n");
for (i = 0; i < 10; i++)
{
scanf("%s", name[i]);
strcpy(tname[i], name[i]);
}
for (i = 0; i < 9 ; i++)
{
for (j = i + 1; j < 10; j++)
{
if (strcmp(name[i], name[j]) > 0)
{
strcpy(temp, name[i]);
strcpy(name[i], name[j]);
strcpy(name[j], temp);
}
}
}
printf("Sorted names\n");
printf("-----------------------------------------\n");
for (i = 0; i < 10; i++)
{
printf("%s\n",name[i]);
}
getch();
}
```

10. WAP to search a string from the given strings.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
int n,i;
int check = 0;
char name[10][25],search[25];
```

```
printf("Enter the number of strings:\n");
scanf("%d",&n);
printf("Enter name of %d strings:\n");
for(i=0;i<n;i++)
{
scanf("%s", &name[i]);
}
printf("Enter any string to search:\n");
scanf("%s",search);
for(i=0;i<n;i++)
{
if(strcmp(name[i],search)==0)
{
check = 1;
break;
}
}
if(check == 0)
{
printf("string not found\n");
}
else
{
printf("string found\n");
}
getch();
}
```

Exercise 4(G)

1.What is function ? Write advantage of function.

A function is a self-contained subprogram, which is meant to do some specific, well-defined task.

Advantages:

a. Function increases code reusability
b. The length of the source program can be reduced by using functions at appropriate places
c. The program development will be faster
d. The program debugging will be easier
e. Large number of programmer can be involved
f. The program can be developed in short period of time
g. The program can be developed in different places
h. The program can be tested and compiled independently by different member of a programming team

2.What are type of function ? Explain.

Basically, there are two types of functions in C on basis of whether it is defined by user or not.

a. **Library function**: Library functions are the in-built function in C programming system.
   For example:
   main()
   - The execution of every C program starts from this main() function.
   printf()
   - prinf() is used for displaying output in C.
   scanf()
   - scanf() is used for taking input in C.

b. **User defined function:** C allows programmer to define their own function according to their requirement. These types of functions are known as user-defined functions. Suppose, a programmer wants to find factorial of a number and check whether it is prime or not in same program. Then, he/she can create two separate user-defined functions in that program: one for finding factorial and other for checking whether it is prime or not. Its types are as follows:

i. Function with no arguments and no return value
ii. Function with no arguments and return value
iii. Function with arguments but no return value
iv. Function with arguments and return value.

3. Explain the components of function.

The components of function are as:

a. **Function prototype:** Function prototype provides the following information to the computer.

- The type of the value entered
- The name of the function
- The number and the type of the arguments that must be supplied in a function call
  Syntax: return_value function_name(arg1, arg2,..... argn)

b. **Function definition**: A function definition has two principle components.

- Function header or function declaratory
- Body of the function

The first line of function definition is known as function delcarator and is followed by the function body.

Syntax: data_type function_name(data_type arg1, data_type arg2,..... data_type argn)

c. **Call the function or function call:** A function call is specified by the function name followed by the arguments enclosed in parenthesis and terminated by a semicolon.
Syntax: function_name(arg1, arg2,.....argn);

4. What is recursion? Explain with one example.

Recursion can be regarded as the ability of function defining an object in terms of simpler case of itself. The characteristics of recursion function are: function should call itself and problem statement must include a stopping condition.
Example:

```
int sum(int n)
{
        if(n<=0)
                return 0;
        else
                return(n+sum(n-1));
}
```

5.Diferentiate between library and user defined function.

Library functions	User defined functions
These are Predefined functions.	These are the function which r created by user as per his own requirements.
• These are part of header file (such as MATH.h) which is called runtime.	These are part of the program which compile runtime
• In this, id of function is given by developers.	• In this, the name of function id decided by user
• Name of function can't be changed.	• In this name of function can be changed any time
Example : SIN, COS, Power, etc.	Example : fibo, mergeme, etc.

Lab Works 6: User defined functions

1.WAP to find the simple interest using function.

```
#include<stdio.h>
#include<conio.h>
void calc(float x,float y,float z);
void main()
{
float p,t,r;
printf("Enter the value of P, T and R(%) respectively:\n");
scanf("%f%f%f",&p,&t,&r);
calc(p,t,r);
getch();
}
void calc(float x,float y,float z)
```

```
{
float si;
si = (x*y*z)/100;
printf("The Simple Interest is: %f\n",si);
}
```

2.WAP to find greatest number among four different numbers.

```
#include<stdio.h>
#include<conio.h>
void calc(float w,float x,float y,float z);
void main()
{
float a,b,c,d;
printf("Enter the value of 4 numbers:\n");
scanf("%f%f%f%f",&a,&b,&c,&d);
calc(a,b,c,d);
getch();
}
void calc(float w,float x,float y,float z)
{
float largest;
if(w>=x && w>=y && w>=z)
{
largest = w;
}
else if(x>=w && x>=y && x>=z)
{
largest = x;
}
else if(y>=w && y>=x && y>=z)
{
largest = y;
}
else
{
largest = z;
}
printf("The largest number is: %f\n",largest);
}
```

3. WAP to display the multiplication table of given number.

```
#include<stdio.h>
#include<conio.h>
void calc(int x);
void main()
{
```

```
int num;
printf("Enter any number less than 10:\n");
scanf("%d",&num);
calc(num);
getch();
}
void calc(int x)
{
printf("The multiplication table of %d is:\n",x);
printf("----------------------------------------\n" ;);
for(int i=1;i<=10;i++)
{
int value = x * i;
printf("%d x %d = %d\n",x,i,value);
}
}
```

4.WAP using user defined function to calculate y raise power to x.

```
#include<stdio.h>
#include<conio.h>
int calc(int y,int x);
void main()
{
int num,power,value;
printf("Enter the value of y and its power x respectively:\n");
scanf("%d%d",&num,&power);
value = calc(num,power);
printf("The value is: %d\n",value);
getch();
}
int calc(int y,int x)
{
if(x ==0)
{
return(1);
}
else
{
return(y*calc(y,x-1));
}
}
```

5.WAP to find factorial of a number using recursion.

```
#include<stdio.h>
#include<conio.h>
int calc(int x);
```

```
void main()
{
int num,value;
printf("Enter the value to calculate its factorial:\n");
scanf("%d",&num);
value = calc(num);
printf("The factorial is: %d\n",value);
getch();
}
int calc(int x)
{
if(x ==0)
{
return(1);
}
else
{
return(x*calc(x-1));
}
}
```

6. WAP to display Fibonacci series using recursion.

```
#include<stdio.h>
#include<conio.h>
int Fibonacci(int);
void main()
{
int n,i = 0,c;
printf("Enter the number of terms of series:\n");
scanf("%d",&n);
printf("Fibonacci series\n");
for ( c = 1 ; c <= n ; c++ )
{
printf("%d\n", Fibonacci(i));
i++;
}
getch();
}
int Fibonacci(int n)
{
if ( n == 0 )
return 0;
else if ( n == 1 )
return 1;
else
return ( Fibonacci(n-1) + Fibonacci(n-2) );
```

```
}
```

7. WAP to find sum and average of n numbers using array and function.

```
#include<stdio.h>
#include<conio.h>
void calc(float x[100],int count);
void main()
{
int n;
float num[100];
printf("Enter the number of values to input:\n");
scanf("%d",&n);
printf("Enter the values one by one:\n");
for(int i=0;i<n;i++)
{
scanf("%f",&num[i]);
}
calc(num,n);
getch();
}
void calc(float x[],int count)
{
float sum=0,avg=0;
for(int i=0;i<count;i++)
{
sum = sum + x[i];
}
avg = sum/count;
printf("The sum is: %f\n",sum);
printf("The average is: %f\n",avg);
}
```

8.WAP to print greatest number among n number using array and function.

```
#include<stdio.h>
#include<conio.h>
void calc(float x[100],int count);
void main()
{
int n;
float num[100];
printf("Enter the number of values to input:\n");
scanf("%d",&n);
printf("Enter the values one by one:\n");
for(int i=0;i<n;i++)
{
scanf("%f",&num[i]);
```

```
}
calc(num,n);
getch();
}
void calc(float x[],int count)
{
float largest=x[0];
for(int i=1;i<count;i++)
{
if(x[i]<x[i+1])
{
largest = x[i+1];
}
}
printf("The largest value is: %f\n",largest);
}
```

9.WAP to print transpose of matrix(2x2 type) using array and function.

```
#include<stdio.h>
#include<conio.h>
void calc(int x[10][10],int rowcount,int colcount);
void main()
{
int m, n, c, d, matrix[10][10];
printf("Enter the number of rows and columns of matrix:\n");
scanf("%d%d",&m,&n);
printf("Enter the elements of matrix:\n");
for( c = 0 ; c < m ; c++ )
{
for( d = 0 ; d < n ; d++ )
{
scanf("%d",&matrix[c][d]);
}
}
calc(matrix,m,n);
getch();
}
void calc(int x[10][10],int rowcount,int colcount)
{
int c,d,transpose[10][10];
for( c = 0 ; c < rowcount ; c++ )
{
for( d = 0 ; d < colcount ; d++ )
{
transpose[d][c] = x[c][d];
}
```

```
}
printf("Transpose of entered matrix:\n");
for( c = 0 ; c < colcount ; c++ )
{
for( d = 0 ; d < rowcount ; d++ )
{
printf("%d\t",transpose[c][d]);
}
printf("\n");
}
}
```

10. WAP to print sum of two matrix (2x2 type) using array and function.

```
#include<stdio.h>
#include<conio.h>
void read_arr(int a[10][10],int row,int col)
{
int i,j;
printf("Enter Elements of matrix:\n");
for(i=1;i<=row;i++)
{
for(j=1;j<=col;j++)
{
scanf("%d",&a[i][j]);
}
}
}

void add_arr(int m1[10][10],int m2[10][10],int m3[10][10],int row,int col)
{
int i,j;
for(i=1;i<=row;i++)
{
for(j=1;j<=col;j++)
{
m3[i][j] = (m1[i][j] + m2[i][j]);
}
}
}

void print_arr(int m[10][10],int row,int col)
{
int i,j;
printf("The sum of two matrices is:\n");
for(i=1;i<=row;i++)
{
```

```
for(j=1;j<=col;j++)
{
printf("%d\t",m[i][j]);
}
printf("\n");
}
}

void main()
{
int m1[10][10],m2[10][10],m3[10][10],row,col;
clrscr();
printf("Enter number of rows :\n");
scanf("%d",&row);
printf("Enter number of colomns :\n");
scanf("%d",&col);
read_arr(m1,row,col);
read_arr(m2,row,col);
add_arr(m1,m2,m3,row,col);
print_arr(m3,row,col);
getch();
}
```

Exercise 4(H)

1.Explain structure and union with respective examples.

Structure is collection of heterogeneous data items treated as a single unit. Each data item is called member of structure. The keyword **struct** is used to declare structure variable and type of structure variable is defined by the **tag_name** of the structure.

Syntax:

```
struct tag_name
{
        data_type member1;
        data_type member2;
        ..
        ..
        ..
        data_type memberN;
}
struct tag_name var1,var2, var3, .., varM;
```

Example:

```
struct student
{
```

```
        int roll;
        char name[10];
        char address[20];
}
Struct student S1, S2, S3;
```

Union is similar to structure but is differs only in its storage location. The union keyword is used to define union. In structure, each member has its own memory block whereas all members of union can share the same memory location. Therefore, union can take less amount of memory than structure and it can access only one member at a time.

Syntax:

```
union tag_name
{
        data_type member1;
        data_type member2;
        ..
        ..
        ..
        data_type memberN;
}
union tag_name var1,var2, var3, .., varM;
```

Example:

```
union data
{
        char x;
        int y;
        float z;
}
var;
```

2.Diffrentiate between structure and union.

Structure	Union
It is the collection of data items of dissimilar data types	Same as structure but it differs in its storage class
Memory size is determined by sum of memories allocated to all the members	Memory size is determined by the memory allocated to the largest member
Multiple members can be simultaneously accessed at a time	Only one member can be accessed at a time.
The struct keyword is used to define structure	The union keyword is used to define union
Syntax:	Syntax:

struct tag_name
{
data_type member1;
data_type member2;
..
..
..
data_type memberN;
}
	union tag_name
	{
	data_type member1;
	data_type member2;
	..
	..
	..
	data_type memberN;
	}

Example:
struct student
{
int roll;
char name[10];
char address[20];
}
Struct student S1, S2, S3;
	Example:
	union data
	{
	char x;
	int y;
	float z;
	}
	var;

3. **Differentiate between array and structure.**

Array	Structure
It is the collection of data items having similar data types	It is the collection of data items having dissimilar data types.
Each data type item is called elements	Each data item is called member
It is built in data type	It is user defined data type
It is not possible to define array of array or structure of array	It is possible to define array of structure

Syntax:
data_type array_name[size];
	Syntax:
	struct tag_name
	{
	data_type member1;
	data_type member2;
	..
	..
	..
	data_type memberN;
	}

Example: Int x[10];	Example: struct student

	{
	int roll;
	char name[10];
	char address[20];
	}
	Struct student S1, S2, S3;

4. Is it possible to pass structure variable to function? Explain with suitable example.

A function can be called by passing structure variable as parameter. Structure variable can be passed in various ways: member can be passed individually or entire structure can be passed at once. The following example can take a structure variable length and calculate area of square of a square.

```
#include<stdio.h>
void calc(struct area length);
struct area
{
        int len;
};
void main()
{
        struct area length;
        length.len = 5;
        calc(length);
}
void calc(struct area length)
{
        int area = length.len*length.len;
        printf("Area is= %d"Â□, area);
}
```

5. What is nested structure? Explain with suitable example.

Nested structure can be defined as the structure as a member of another structure.

Example:

```
struct date
{
        int day;
        int month;
        int year;
};
struct student
{
        int roll;
        char fname[30];
        char lname[30];
        struct date d;
```

```
};
void main()
{
struct student s;
s.roll = 10;
s.fname = "ABC"Â□;
s.lname = "XYZ"Â□;
s.d.year = 2014;
s.d.month = 9;
s.d.day = 18;
printf("Roll: %d\n First name: %s\n Last name: %s\n Date of Birth: %d/%d/%d"Â□,
s.roll,s.fname,s.lname,s.d.year,s.d.month,s.d.day"Â□);
}
```

6."Structure is called user defined data type". Justify your answer with example.

The typedef keyword allows us to define new data types that are equivalent to existing data types. This feature also can be implemented to structure to define user defined data types

Example:

```
typedef struct
{
int feet;
int inch;
}
Distance d1, d2, d3;
```

In the example above, Distance is user defined data type and d14, d2, d3 are the structure variables with data type Distance.

Lab Works 7: Structure and Union

1.WAP considering user defined variable distance with members: feet and inch. Calculate the sum two different distance.

```
#include<stdio.h>
#include<conio.h>
struct distance
{
float feet,inch;
}d[3];
void main()
{
printf("Enter the distance-1 in feet and inches respectively:\n");
scanf("%f%f",&d[0].feet,&d[0].inch);
printf("Enter the distance-2 in feet and inches respectively:\n");
scanf("%f%f",&d[1].feet,&d[1].inch);
d[2].feet = d[0].feet + d[1].feet;
d[2].inch = d[0].inch + d[1].inch;
```

```
printf("The sum is: %f feet and %f inch\n",d[2].feet,d[2].inch);
getch();
}
```

2.WAP considering user defined variable time with members :hh, mm, and ss, calculate the sum of two different times.

```
#include<stdio.h>
#include<conio.h>
struct time
{
int hh,mm,ss;
}t[3];
void main()
{
printf("Enter the time-1 in HH, MM and SS respectively:\n");
scanf("%d%d%d",&t[0].hh,&t[0].mm,&t [0].ss);
printf("Enter the time-2 in HH, MM and SS respectively:\n");
scanf("%d%d%d",&t[1].hh,&t[1].mm,&t [1].ss);
t[2].hh = t[0].hh + t[1].hh;
t[2].mm = t[0].mm + t[1].mm;
t[2].ss = t[0].ss + t[1].ss;
printf("The sum is: %d HH, %d MM and %d SS\n",t[2].hh,t[2].mm,t[2].ss);
getch();
}
```

3. Consider structure with members : sid, name and height.WAP that takes maximum 100 records, rearrange the record in ascending order on the basis of height and print them.

```
#include<stdio.h>
#include<conio.h>
struct student
{
int sid,height;
char name[25];
}s[100];
void main()
{
int n;
printf("Enter the number of students:\n");
scanf("%d",&n);
if(n>100)
{
printf("Maximum allowed is 100 only\n");
}
else
{
```

```
printf("Enter the sid,name and height of %d students respectively:",n);
for(int i=0;i<n;i++)
{
printf("\nEnter sid:");
scanf("%d",&s[i].sid);
printf("Enter name:");
scanf("%s",s[i].name);
printf("Enter height:");
scanf("%d",&s[i].height);
}
for(i=0;i<n;i++)
{
for(int j=0;j<n-1;j++)
{
if(s[j].height>s[j+1].height)
{
struct student temp;
temp.sid = s[j+1].sid;
*temp.name = *s[j+1].name;
temp.height = s[j+1].height;

s[j+1].sid = s[j].sid;
*s[j+1].name = *s[j].name;
s[j+1].height = s[j].height;

s[j].sid = temp.sid;
*s[j].name = *temp.name;
s[j].height = temp.height;
}
}
}
printf("\nThe sorted records are:\n");
printf("-------------------------\n");
for(i=0;i<n;i++)
{
printf("sid: %d\n",s[i].sid);
printf("name: %s\n",s[i].name);
printf("height: %d\n\n",s[i].height);
}
}
getch();
}
```

4.WAP that takes empid,name and salary of 100 employee and search a particular employee's record in the array of structure on the basis of empid. Also find the position of the record.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
struct employee
{
int empid,salary;
char name[25];
}e[100];
void main()
{
int searchid;
printf("Enter the empid,name and salary of 100 employees respectively:");
for(int i=0;i<100;i++)
{
printf("\nEnter empid:");
scanf("%d",&e[i].empid);
printf("Enter name:");
scanf("%s",e[i].name);
printf("Enter salary:");
scanf("%d",&e[i].salary);
}
printf("\nEnter the empid to search:");
scanf("%d",&searchid);
for(i=0;i<100;i++)
{
if(e[i].empid == searchid)
{
printf("\nposition: %d, empid: %d, name: %s, salary:
%d\n",(i+1),e[i].empid,e[i].name,e[i].salary);
break;
}
}
getch();
}
```

5. Define structure type employee with members: empid, name , post and salary. WAP to print the record of the employee who has maximum salary.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
struct employee
{
int empid,salary;
char name[25],post[25];
}e[100];
void main()
```

```
{
int greatest,id;
printf("Enter the empid,name and salary of 100 employees respectively:");
for(int i=0;i<3;i++)
{
printf("\nEnter empid:");
scanf("%d",&e[i].empid);
printf("Enter name:");
scanf("%s",e[i].name);
printf("Enter post:");
scanf("%s",e[i].post);
printf("Enter salary:");
scanf("%d",&e[i].salary);
}
greatest = e[0].salary;
id = 0;
for(i=1;i<3;i++)
{
if(e[i].salary > greatest)
{
greatest = e[i].salary;
id = i;
}
}
printf("\nThe greatest salary is of employee:\n");
printf("position: %d\n", id);
printf("empid: %d\n", e[id].empid);
printf("name: %s\n", e[id].name);
printf("post: %s\n", e[id].post);
printf("salary: %d\n", e[id].salary);
getch();
}
```

Exercise 4(I)

1.Define pointer and its uses.

A pointer is a variable that points to another variable which means it contains the memory address of another variable and is declared as the pointer type.

The uses of pointers are as follows:

a. Passing parameter by reference
b. Accessing array element
c. Dynamic memory allocation
d. Reducing size of parameter

2.Differentiate between array and pointer.

Pointer	Array
1 . A pointer is a place in memory that keeps address of another place inside	1 . An array is a single, pre allocated chunk of contiguous elements (all of the same type), fixed in size and location.
2. Pointer can't be initialized at definition.	2. Array can be initialized at definition. Example int num[] = { 2, 4, 5}
3 . Pointer is dynamic in nature. The memory allocation can be resized or freed later.	3 . They are static in nature. Once memory is allocated, it cannot be resized or freed dynamically.
4 . The assembly code of Pointer is different than Array	4 . The assembly code of Array is different than Pointer.

3.Write down the similarities and differences of array with pointer.

Similarities:

Arrays are defined as const pointer. Array and pointer usage notation can be interchange.

Pointer can be used in array notation:

```
char* ptr=new char[3];
ptr[0]='a';
ptr[1]='b';
```

Array can be used in pointer notation:

```
char arr[]="abcde";
*(arr+4)='d';
```

Differences:

Only for pointers, you need to explicitly allocate and deallocate memory, unless pointing to variable. This will be in heap unlike stack as in case for arrays.

For pointer, memory to which it points can be resized using realloc. Array cannot be resized.

You can take address of the pointer variable. Address of array will give you address for first element of that array. This is because array is just a name to a set of memory locations, whereas pointer is a variable.

Sizeof pointer is always fixed. 8 byte for 64 bit machines and 4 bytes for 32 bit machine to store the start address (64 bit or 32 bit) of memory. Sizeof array is strlen of array + 1.

4.Define the term call by value and call by references.

If data is passed by value, the data is copied from the variable used in for example main() to a variable used by the function. So if the data passed (that is stored in the function variable) is modified inside the function, the value is only changed in the variable used inside the function

If data is passed by reference, a pointer to the data is copied instead of the actual variable as is done in a call by value. Because a pointer is copied, if the value at that pointers address is changed in the function, the value is also changed in main().

5.Differentitae between structure and pointer with example.

Structure	Pointer
It is the collection of data items of dissimilar data types	A pointer is a variable that points to another variable which means it contains the memory address of another variable and is declared as the pointer type
Memory size is determined by sum of memories allocated to all the members	Dynamically memory allocates and also releases
The struct keyword is used to define structure	* operator is placed before any variable declaration

Syntax:
struct tag_name
{
data_type member1;
data_type member2;
..
..
..
data_type memberN;
}
	Syntax:
	Data_type *variable_name;

Example:
struct student
{
int roll;
char name[10];
char address[20];
}
Struct student S1, S2, S3;
	Example:
	char *p;
	int *x;

6. Define the indirection and address of operators in pointers.

An indirection operator, is an operator used to obtain the value of a variable to which a pointer points. While a pointer pointing to a variable provides an indirect access to the value of the variable stored in its memory address, the indirection operator dereferences the pointer and returns the value of the variable at that memory location. The indirection operator is a unary operator represented by the symbol (*).

An address-of operator is a mechanism that returns the memory address of a variable. These addresses returned by the address-of operator are known as pointers, because they "point" to the variable in memory. The address-of operator is a unary operator represented by an ampersand (&). It is also known as an address operator.

Lab Works 8: Pointer.

1. WAP to swap two numbers using pointer.

```
#include<stdio.h>
#include<conio.h>
void swap(int *a,int *b);
void main()
{
int a = 20;
int b = 30;
printf("Before swapped: a = %d and b = %d\n",a,b);
swap(&a,&b);
printf("Swapped result: a = %d and b = %d\n",a,b);
getch();
}
void swap(int *a, int *b)
{
int temp;
temp = *a;
*a = *b;
*b = temp;
}
```

2.WAP using call by value and call by refrences.

Using call by value:

```
#include<stdio.h>
#include<conio.h>
void swap(int,int);
void main()
{
int a = 20;
int b = 30;
printf("Before swapped: a = %d and b = %d\n",a,b);
swap(a,b);
getch();
```

```
}
void swap(int a, int b)
{
int temp;
temp = a;
a = b;
b = temp;
printf("Swapped result: a = %d and b = %d\n",a,b);
}
```

Using call by reference:

```
#include<stdio.h>
#include<conio.h>
void swap(int *a,int *b);
void main()
{
int a = 20;
int b = 30;
printf("Before swapped: a = %d and b = %d\n",a,b);
swap(&a,&b);
printf("Swapped result: a = %d and b = %d\n",a,b);
getch();
}
void swap(int *a, int *b)
{
int temp;
temp = *a;
*a = *b;
*b = temp;
}
```

3.WAP to input 25 numbers and numbers and display them ascending order.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int num[25],*p;
printf("Enter 25 numbers:\n");
for(int i=0;i<25;i++)
{
scanf("%d",&num[i]);
}
p = num;
for (i = 0; i < 25; ++i)
{
for (int j = i + 1; j < 25; ++j)
```

```
{
if (p[i] > p[j])
{
int a = p[i];
p[i] = p[j];
p[j] = a;
}
}
}
printf("The ordered numbers are:\n");
printf("----------------------\n");
for(i=0;i<25;i++)
{
printf("%d\n",p[i]);
}
        getch();
}
```

4. WAP to input 100 numbers and search a given number given by yuser.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int num[100],*p,search,check=0;
printf("Enter 100 numbers:\n");
for(int i=0;i<100;i++)
{
scanf("%d",&num[i]);
}
p = num;
printf("Enter a number to search:");
scanf("%d",&search);
for(i=0;i<100;i++)
{
if(p[i] == search)
{
check = 1;
break;
}
}
if(check == 0)
{
printf("Number Not Found\n");
}
else
{
```

```
printf("Number Found\n");
}
getch();
}
```

5.WAP to calculate the sum and the average of 10 numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int num[10],*p,sum=0;
float avg;
printf("Enter 10 numbers:\n");
for(int i=0;i<10;i++)
{
scanf("%d",&num[i]);
}
p = num;
for(i=0;i<10;i++)
{
sum = sum + p[i];
}
avg = sum/10;
printf("Enter sum is: %d\n",sum);
printf("The average is: %f\n",avg);
getch();
}
```

Exercise 4(J)

1.What is data file? Write the syntax to open and close file.

A data file is collection of data or information which is permanently stored inside secondary memory.

Syntax:

Open data file:

```
FILE *fp;
Fp = fopen("filename"Â□, "mode"Â□);
```

Close data file:

```
fclose(filepointer);
```

2. Define the term: FILE and EOF.

The keyword FILE is used to define file data type. It's a data type defined in the ANSI C standard to operate with files. It usually points to an internal structure that describes the file and its current state to the library functions.
EOF is used to represent end of file. EOF marks are helpful in data transmission and storage. Files are stored in blocks, and the end marker helps the computer know it has allocated enough space to store the file.

3.Differentitae between sequential access and random access technique of data file.
In sequential file processing, data is sequentially accessed from/to data file. It is very time consuming technique. If we need to access the last record of a file, then we need we need to access all the records before that record.
In random file processing, data is randomly accessed anywhere from/to data file. It is more faster technique than the previous one. We can read and write data in particular part of a file with the help of the functions ftell(), fseek(), rewind().

4. What are the file input/ output function? Explain with their systax.
The popular input output functions are as follows:

a. getc()/putc():
   Function getc() helps us to read single character from file and putc() helps us to write single character to file.
   Syntax:
   character_variable = getc(file_pointer);
   putc(character_variable.file_pointer) ;
b. getw()/putw:
   Function getw() helps us to read single integer from file and putw() helps us to write single integer to the data file.
   Syntax:
   integer_variable = getw(file_pointer);
   putw(integer_variable.file_pointer);< /FONT>
c. fscanf()/fprintf():
   Function fscanf() helps us to read formatted data from file and fprintf() helps us to write formatted data to file.
   Syntax:
   fscanf(file_pointer, "control string"Â□, variables);
   fprintf(file_pointer, "control string"Â□, variables);
d. fread()/fwrite():
   Function fread() helps us to read structure type data i.e. record form data file and function fwrite() helps us to write structure type data to the data file.
   Syntax:
   fread(&variable, sizeof(tagname), number_of_record, file_pointer);
   fwrite(&variable, sizeof(tagname), number_of_record, file_pointer);

5.Write the functions of rename() and remove () with examples.
The remove() function helps us to delete this file specified as a parameter and the general syntax is:
remove("filename"Â□);

The rename() function helps us to rename the existing file into new file and the general syntax is:
rename("oldfilename"Â□, "newfilename"Â□);
Example:

```
#include<stdio.h>
void main()
{
FILE *fp;
char name[30];
fp = fopen["oldfile.txt"Â□, "w"Â□];
printf("\nEnter name : "Â□);
scanf("%s"Â□, name);
fprintf(fp, "%s"Â□, name);
fclose(fp);
rename("oldfile.txt"Â□, "newfile.txt"Â□);
remove("newfile.txt"Â□);
}
```

Lab Works 9: Working with files.

1.WAP to enter name, roll_no and marks of 10 students and store them in the file.

```
#include<stdio.h>
#include<conio.h>
struct student
{
char name[25];
int roll,marks;
}s[10];
void main()
{
FILE *fp;
fp = fopen("test.txt","w");
printf("Enter the name, roll and marks of 10 students respectively:\n");
for(int i=0;i<10;i++)
{
scanf("%s%d%d",s[i].name,&s[i].roll,&s [i].marks);
fprintf(fp,"%s\t%d\t%d\n",s[i].name,s[i].roll ,s[i].marks);
}
fclose(fp);
getch();
}
```

2.WAP to store empid, name and salary of 10 employees in a data file and display the records from the file.

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

```
struct employee
{
int empid,salary;
char name[25];
}e[10];
void main()
{
FILE *fp;
fp = fopen("test.txt","w");
printf("Enter the empid, name and salary of 10 employees respectively:\n");
for(int i=0;i<10;i++)
{
scanf("%d%s%d",&e[i].empid,e[i].name,& ;e[i].salary);
fprintf(fp,"%d\t%s\t%d\n",e[i].empid,e[i].name ,e[i].salary);
}
fclose(fp);
fp = fopen("test.txt","r");
printf("The records in file are:\n");
i = 0;
while((fscanf(fp,"%d\t%s\t%d",&e[i].empid,e [i].name,&e[i].salary)) != EOF)
{
printf("empid: %d, name: %s and salary: %d\n",e[i].empid,e[i].name,e[i].salary);
i++;
}
fclose(fp);
getch();
}
```

3.WAP that reads successive records from new data file and display each record in the screen in an appropriate format.

Same Above.

4.WAP to store std_id, name and mark of n students in a data file. Display the records appropriate format reading from the file.

```
#include<stdio.h>
#include<conio.h>
struct student
{
int roll,marks;
char name[25];
}s[10];
void main()
{
int n;
printf("Enter the number of records you want to enter:");
scanf("%d",&n);
```

```
FILE *fp;
fp = fopen("test.txt","w");
printf("Enter the name, roll and marks of %d employees respectively:\n",n);
for(int i=0;i<n;i++)
{
scanf("%s%d%d",s[i].name,&s[i].roll,&s [i].marks);
fprintf(fp,"%s\t%d\t%d\n",s[i].name,s[i].roll ,s[i].marks);
}
fclose(fp);
fp = fopen("test.txt","r");
printf("The records in file are:\n");
i = 0;
while((fscanf(fp,"%s\t%d\t%d",s[i].name,&s [i].roll,&s[i].marks)) != EOF)
{
printf("name: %s, roll: %d and marks: %d\n",s[i].name,s[i].roll,s[i].marks);
i++;
}
fclose(fp);
getch();
}
```

5. WAP to enter name, roll, roll_no and marks of 10 students and store them in the file. Read and displays the same from the file.

```
#include<stdio.h>
#include<conio.h>
struct student
{
char name[25];
int roll,marks;
}s[10];
void main()
{
FILE *fp;
fp = fopen("test.txt","w");
printf("Enter the name, roll and marks of 10 students respectively:\n");
for(int i=0;i<10;i++)
{
scanf("%s%d%d",s[i].name,&s[i].roll,&s [i].marks);
fprintf(fp,"%s\t%d\t%d\n",s[i].name,s[i].roll ,s[i].marks);
}
fclose(fp);
fp = fopen("test.txt","r");
printf("The records in file are:\n");
i = 0;
while((fscanf(fp,"%s\t%d\t%d",s[i].name,&s [i].roll,&s[i].marks)) != EOF)
{
```

```
printf("name: %s, roll: %d and marks: %d\n",s[i].name,s[i].roll,s[i].marks);
i++;
}
fclose(fp);

getch();
}
```