Contents

INTRODUCTION TO 'C' PROGRAMMING

- C is a general purpose programming language.
- In 1972 Dennis Ritchie at bell labs wrote C.
- In 1983 the American National Standards Institute (ANSI) established a committee to provide a comprehensive definition of C. The resulting definition "ANSI C" was completed in 1988.
- C is a structure programming language. It is also called a middle level language, because it supports the feature of memory management, the feature of conditional, loop statements (Supports High Level Language) etc.
- Since, C pointers (Direct Memory Access-DMA) and supports low level and high level languages, it is used for application development as well as system programming.

(C AND C++) PROGRAMMING BASICS

1. CONSTANT

The input and output data's in a program

Numeric Constant	**Non Numeric (String) constant**
Without decimal Int, short int, long int Ex:100,500,10000 **With decimal** float , double Ex:4.53, 3.4, 4.88	Data's within single Quote or double quote marks. Does not involve in Arithmetic Process. Single char 'a' Multiple char "raja"

2. VARIABLE DECLARATION

- Space allocation to variable.
- Data types are used to allocate required space.
- Every variable must declare before it is used.

Syntax to declare a variable
DataType Variable name [=value];

Note: The square brackets imply that the particular part of the syntax is optional.

DataTypes:

int	2/4 bytes
short int	1 byte
long int	4 bytes
float	4 bytes
double	8 bytes
char	1 byte

Example:

```
 int a;
int a,b,c;
int a=10; b=20; c;
char ch, na;
```

3. VARIABLE

- Computer memory is used to store all input data's.
- Memory has collection of cells and each cell has an address.
- Address cannot read by the user. Hence an alias name is used to identify the location. This alias name is called variable.

For Example :

int a1;
Allocate 4 bytes to a1 and the first address of the allocated byte assign to a1.
a1 = 10; stores the value 10 into the memory location.

4. C OPERATORS AND EXPRESSIONS

- Expressions in C are built from combination of Operators and Operands.
- Operands are either variable or data.
- Generally there are three types of expressions.

a. Arithmetic Expression
b. Relational Expression
c. Logical Expression

A. Arithmetic Expression

- Used to do arithmetic process.
- Computes operands and return a result.
- Results will store in a variable or will display.

A.1. Arithmetic Operators

+ - * / %

5 / 2 = 2 - if both data is integer the result is the integer value
5 % 2=1 - % return the remainder value.

A.2. Unary Operators (+ or -)

Example: x = - y; Here negative of the y value will be stored in x.

A.3. Increment (++) / Decrement (--) Operators

The value of the operand incremented by one.

Example: a=5;
a++; then a=6
or
++a;
a++ - postfix increment
++a - prefix increment

Note:
Example 1: a=5;
b=a++;
then b=5, a=6

Example 2: a=5;
b= ++a;
then b=6,a=6;

A.4. Assignment Operator (=)

Variable=<expression>;

Example 1: a=5; b = a; c = a + b;
then c=10

Example 2: a=b=c=5;

A.5. Compound Assignment Operator

Variable<operator>=value;

a=5;
a+=2; then a=7;
a+=3; then a = 10;

We may have

+= -= += /= %= ^=

The Procedure of Operators:

The priority of the operators following binomial rules

1 () or [] 2 ^ 3 * or / 4 + or -

B. Relational Expression

Compare two values using relational operator and return true (1) or false (0).

Relational Operators:

< > < = > = == ! =

Example : a=5 b=10 c=5

a > c	false
a == c	true
b <= (a+b)	true
b >c	true
(c+a) == b	true

C. Logical Expression

Logical Expression compares two or more relations and returns the result as true (1) or false (0).

Logical Operators:

&& (and) || (or) ! (not)

Example - 1:

To find the result of a student with three subject marks as pass or fail

t >=35 && e >=35 && m >=35

In the above expression if all the relation is true then it returns 1 (true) or 0 (false)

Example - 2:

!true = false and !false = true

D. Conditional Operator

exp1 ? exp2 : exp3;

if exp1 is true value return will be exp2 otherwise exp3.

Example:

```
int x,y,z;
z = (x>y)?x:y;
```

If the condition is true then z=x otherwise z=y

E. Cast Operator

Data type of any expression or identifier can be converted into another type using explicit casting format

Example:

int x; float y; y=5;
C – PROGRAM x=int(y);
C++ PROGRAM x = (int)y;

Here the float value y is converted into integer x.

Note: type conversion is not required when a small data stores in large data type

F. Sizeof Operator

This return number of bytes the operand occupies in memory.

Example:

```
sizeof(int) = 2 bytes
int a;
sizeof(a) = 2 bytes

struct stu
{
        long int rollno;
        char na[20];
        int t, e, m;
};
sizeof(struct stu) = 30 bytes
```

G. Bitwise Shift Operators

The bitwise shift operators take two operands: the first is a quantity to be shifted, and the second specifies the number of bit positions by which the first operand is to be shifted. The direction of the shift operation is controlled by the operator used.

Shift operators convert their operands to 32-bit integers in big-endian order and return a result of the same type as the left operand. The right operand should be less than 32, but if not only the low five bits will be used.

<< (Left shift)

This operator shifts the first operand the specified number of bits to the left. Excess bits shifted off to the left are discarded. Zero bits are shifted in from the right.

For example : 9 << 2 yields 36:
9 (base 10):
00001001 (base 2)
9 << 2 (base 10):
00100100 (base 2) = 36 (base 10)

>> (Sign-propagating right shift)

This operator shifts the first operand the specified number of bits to the right. Excess bits shifted off to the right are discarded. Copies of the leftmost bit are shifted in from the left. Since the new leftmost bit has the same value as the previous leftmost bit, the sign bit (the leftmost bit) does not change. Hence the name "sign-propagating".

For example : 9 >> 2 yields 2:
9 (base 10):
00001001 (base 2)
9 >> 2 = 2 (base 10)
00000010 (base 2)

More Examples:

Mathematical Expression	**C Equivalent Expression**
$\frac{X+y}{Z}$	(x+y)/z
$\frac{(a+b)^2}{(a-b)^2}$	Pow((a+b),2) / pow((a+b),2)

C OPERATORS AND ITS PRECEDENCE

Parentheses, braces	**(),[]**
Unary Operators	**-,++,--,!,~,&**
Multiplicative Operators	***, /,%**
Additive Operators	**+, -**
Shift Operators	**<<, >>**
Relational Operators	**<, <=, >, >=**
Equality Operators	**==, !=**
Bitwise Operators	**&, ^, |**
Logical Operators	**&&, ||**
Conditional Operators	**?:**
Assignment Operators	**=, +=,-=,*=,/=,%=,&=,^=,|=,<<=, >>=**
Comma Operators	**.**

Example - 1:

Program for all type of operators

Note : in C false = 0, true = 1 and above

```
#include <stdio.h>
Void main()
{
        Int n1,n2,temp;
        Int sum, diff, prod, quotient, remainder;
        Int a=3,b=6,c;
        Float t;
        N1 = 5;
        N2 = 3;
        Sum = n1 + n2;
        Diff = n1 – n2;
        Prod = n1 * n2;
        Quotient = n1 / n2;
        Remainder = n1 % n2;
        Printf("sum = %d\n",sum);
        Printf("Difference = %d\n",diff);
        Printf("Product = %d\n",prod);
        Printf("Quotient = %d\n",quotient);
        Printf("remainder = %d\n",remainder);
        Printf("a = %d\n",a++);
        Printf("b = %d\n",++b);
        Printf(" a = %d\n",a--);
        Printf("b = %d\n",--b);
        A= 2, b = 3;
        If ( a == b)
         Printf("a and b are equal\n");
        Else
         Printf("a and b are not equal \n");
        C = a && b;
        B = a || b || c;
        A = a && b || c;
        Printf("Logical result a = %d\tb =%d\tc=%d\n",a,b,c);
        A= 5, b = 10;
        C = a > b?a:b;
        Printf("the result of c = %d\n",c);
        B = (a = 20, a + 10);
        Printf("a = %d and b = %d (before)\n",a,b);
        Temp = a, a=b,b=temp;
        Printf("a = %d and b = %d (after)\n",a,b);
        Printf("size of a = %d\n",sizeof(a));
        Printf("size of b = %d\n",sizeof(b));
        Printf("size of char = %d\n",sizeof(char));
        Printf("size of double = %d\n",sizeof(double));
        A = 5, b = 10, c = 1;
        A += b;
        B -= a;
        Z *= a;
        Printf("a = %d\n",a);
        Printf(" b = %d\n",b);
        Printf("c =%d\n",c);
        A = 2
        B = 1/ a;
        Printf("b = %d\n",b);
        T= 1/(float)a;
        Printf("t = %f\n",t)
        A = 9;
        A = sqrt(a);
        Printf("square root of a = %d\n",a);
        A = -9;
        A = abs(a);
        Printf("absolute of a = %d\n",a);
        A = 4;
        B = 2;
        C = pow(a,b);
        Printf("a to the power of b = %d\n",c);
        Getch();
}
```

5. C PROGRAMMING STRUCTURE

The structure of C program is

- Comments Statement - (option)

  C - /* message */
  C++ /* */ and // Message

- Inclusion of Header Files - (option)

  C – LANGUAGE

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

  C++

```
#include <iostream.h>
```

- Directives Declaration - (option)

```
#define max 100
#define sqr(x) x*x
```

- Global Variable or Function Declarations - (option)
- Main() Function - (must)
- Sub Function Declaration - (option)

6. STATEMENTS

6.1. Assignment Statement

Variable = value;

```
a = 10;
b = a;
c = a + b;
d =sqr(4);
```

6.2. Input Statement

scanf("conversionstring",variablelist);

- Variable Names prefix with & except string variable.
- Two or more variables separated by comma
- Conversion String define the variable format

%d	-	int
%ld	-	long int
%f	-	float
%lf	-	double
%c	-	char
%s	-	string
%u	-	Unsigned int
%g	-	float
%h	-	read a short int number
%i	-	read a decimal or hexadecimal or octal
%o	-	read an octal number
%p	-	read a pointer
%x	-	read a hexadecimal
%u	-	read an unsigned int

Example :

```
int a,b,c;
scanf("%d%d%d",&a,&b,&c);

int a; float b; char ch; char na[20];
scanf("%d%f%c%s",&a,&b,&ch,na);
```

C++ Input Satement

cin >> v1 >> v2 >> v3….;

to input 2 integer values in c++

```
int a, b;
cin >> a >> b;
```

input name and age in c++

```
char na[20];
int age;

cin >> na >> age;
```

6.3. Output Statement

printf("conversionstring",variablelist);

Conversion String

1. Any Text Message – to display
2. Control Characters - \n, \t
3. Variable Formats - %d, %ld, %f, %lf, %c, %s, %U

Example :

```
printf("enter a value");
printf("slno\tname\taddress");
        slno    name   address
printf("raman\nkannan\nashok");
        raman
        kannan
        ashok
a = 5
printf(" %d",a);
        ans : 5
a = 5 b = 120
printf("%d factorial value = %d",a,b);
        5 factorial value = 120
printf("raman\'s House");
        raman's house
```

C++ Output Statement

Cout << v1 << v2…;

Example – 1:

```
Cout << "IWS INDIA \n Welcomes U";
```

Output:

```
IWS INDIA
Welcomes U
```

Example - 2:

a = 5
b = 120
cout << a << " factorial value = " << b;

Ans : 5 factorial value = 120

6.4. Decision Making Statement

if...

```
int i = 1;
while (i < 101)
{
        if (i % 7 == 0)
        {
          printf("%d\t",i);
        }
        i++;
}
```

if .... else...

```
int i = 1;
while (i < 101)
{
        if (i % 2== 0)
                printf("Even %d",i);
        else
        {
                printf("odd %d",i);
        }
        i++;
}
```

if....else if...else if...else...

```
int avg;
if (avg >= 80)
        printf("distinction");
else if (avg >= 60)
        printf("I class");
else if (avg >= 50)
        printf("II class");
else
        printf("III class");
```

Conditional Operator

(exp)?true:false;

```
c = (a > b)?a : b;

if (a > b)
        c = a;
else
        c = b;
```

1. **switch... case.... default..**.

```
int i;
switch(i)
{
        case 1:
                printf("one");
                break;
        case 2:
                printf("two");
                break;
        case 3:
                printf("three");
                break;
        default:
                printf("value > 3");
                break;
}
```

6.5. Loop Statement

Format -1

```
while(exp)
{ statements;}
```

Format - 2

```
do
{
        statements;
 }while(exp);
```

Format - 3

```
for(initial; condition; increment)
{ statements;}
```

Example - 1:

```
int i = 1;
while (i < 101)
{
        printf("%d",i);
        i++;
}
```

Example - 2:

```
int i = 1;
do
{
        printf("%d",i);
        i++;
}while(i < 101);
```

Example - 3:

```
int  i;
for(i=1; I <101; i++)
printf("%d",i);
for(i=1,j=1,k=1; i <= 100 && j <=100; i++,j++,k++)
printf("%d",j);
int  i = 1;
for(;i < 11;)
{
        i++;
}
```

Example - 4:

```
int  i = 1;
for (;;)
{
        printf("%d",i);
        i++;
        if ( i == 10) break;
}
```

Example - 5:

/* N -multiplication table program */

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

Example – 6:

```
/* display Given value in Reverse Order */
#include <stdio.h>
#include <conio.h>
void main()
{
        int i,j,n;
        printf("Enter a value");
        scanf("%d",&n);
        i = n;
        j = 0;
        while (i > 0)
        {
                j = (j*10) + (i % 10);
                i = i / 10;
        }
        printf("Given value in Reverse order= %d",j);
        if (j == n)
        printf("Given value is a palindrome number");
        else
        printf("Given value is not a palindrome     number");
}
```

6.6. Break and Continue Statements

Break: when it is executed the control of flow jumps to the statement immediately following the end curly brace of the loop statement.

Continue: when it is executed the control of flow jumps to the statement to check the loop condition once again and omit rest of statement up to the end of the block.

Example:

```
#include <stdio.h>
#include <conio.h>
void main()
 int a,b,c;
 while(1)
 {
        printf("Enter 2 values");
        scanf("%d%d",&a,&b);
        printf("1…Add 2..sub 3..divide 4.multiply 5…exit);
        printf("\n Enter ur choice..");
        scanf("%d",&c);
        if(c==5) break;
        if(c<1 || c>5) continue;
```

```
switch( c )
        {
        case 1:
                printf("sum=%d",a+b);
                break;
        case 2:
                printf(sub=%d",a-b);
                break;
        case 3:
                printf("divide=%d",a/b);
                break;
        case 4:
                printf("mul=%d",a*b);
                break;
        }
}
```

6.7. Goto Label STATEMENT

Goto statement is used to change the flow of control in a program without any condition.

Label:

Goto Label:

Example:

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int a;
        a=1;
start:
        printf("a=%d\t",a);
        a++;
        if (a<10) goto start;
}
```

Output: display 1 2 3 … 9

7. ARRAY

7.1 SINGLE DIMENSION

data type variable[index] = {default values};

```
int a[5];
int a[3] = {1,2,3};
a[0] = 1  a[1] = 2 a[2] = 3;
```

Example - 1:

Write a program that input 5 values and display given values in ascending / descending order.

```
#include <iostream.h>
#include <conio.h>
void main()
{
        int a[5],i,j,k;
        cout << "Enter 5 values";
        for(i = 0;i < 5; i++)
        cin >> a[i];
        for(i = 0;i < 4; i++)
        {
                for(j = i + 1; j < 5; j++)
                {
                        if (a[i] > a[j])
                        {
                                k = a[i];
                                a[i] = a[j];
                                a[j] = k;
                        }
                }
        }
        cout << "\n Given values in ascending order";
        for (i=0;i<5;i++)
        cout << "\t" << a[i];
        cout << "\n Given values in Descending order";
        for (i = 4; i>=0;i--)
        cout << "\t" << a[i];
        getch();
}
```

Example -2

Input a string and display each character in that string

```
#include <string.h>
void main()
{
        char na[20];
        int i,j;
        printf("enter a string");
        scanf("%s",na);
        for(i = 0;i<strlen(na);i++)
        printf("%c\n",na[i];
}
```

7.2 TWO DIMENTIONAL ARRAY

datatype variable[rowindex][colindex] = {values};

```
int a[3][2];
int a[3][2] = {1,2,3,4,5,6};
int a[3][2] = {{1,2},{3,4},{5,6}};
for(i = 0; i < 3;i++)
{
        for(j = 0; j <2; j++)
        printf("\t%d",a[i][j]);
}
printf("\n");
char na[20] = "raja";
char na[3][20] = {"raja","bala","ganesh"};
for (i = 0; i < 3; i++)
printf("\t%s",na[i]);
```

String Functions <string.h>

1.strlen(string) - return total number of characters in that string
        strlen("iws") = 3
2. strcpy(s1,s2) : s1 = s2
        copy s2 string to s1.
3.strncpy(s1,s2,n)
        copy first n characters from s2 to s1.
        e.g: strncpy(s1,"ramkumar",3)
        s1 = "ram"
4. strcat(s1,s2) : s1 = s1 + s2;
        Combine both strings and store in s1

5.strncat(s1,s2,n)

Appends first n characters of the second string at the end of the first string
S1= “ram”
S2 = “balakumar”
Strncat(s1,s2,4)
Now s1 = “rambala”;

6. strcmp(s1,s2)

s1 > s2 result > 0
s1 < s2 result < 0
s1 == s2 result == 0

7.strncmp(s1,s2,n)

Compare first n characters from both string and return < 0 or > 0 or == 0

Example:

Strncmp(“ramkumar”,”rambala”,3) == 0

8.Strlwr(string);

Convert any uppercase letters to its equivalent lowercase.

Example:

strlwr(“RAM”) = “ram”

9.Strupr(String);

Covert any lowercase letters to its equivalent uppercase

10.Strupr(“ram”) = “RAM”

11.Strchr(string, desired-char)

This function searches for a specified character in a given string. It returns NULL if the desired character is not found in the string. If the given character is found, then it returns the address of that character.

Example :

strchr(“iws”,”w”)
Return : “ws”

Example – Two Dimensional Arrays

Input 5 names and display given names in ascending order.

```
#include <iostream.h>
#include <conio.h>
void main()
{
        int i,j,k;
        char na[5][20],na1[20];
        cout << "Enter 5 names";
        for(i = 0;i < 5; i++)
        cin >> na[i];
        for(i = 0;i < 4; i++)
```

```
        {
                for(j = i + 1; j < 5; j++)
                {
                        if (strcmp(a[i],a[j]) > 0)
                        {
                                strcpy(na1,na[i]);
                                strcpy(na[i],na[j]);
                                strcpy(na[j],na1);
                        }
                }
        }
        cout << "\n Given Names in ascending order";
        for (i=0;i<5;i++)
        cout << "\t" << na[i];
        getch();
}
```

8. STRUCTURE

1. Define a structure (user defined datatype)

```
struct struct-name
{
        variable decleration;
}[structure variable];
```

2. create a structure variable:

```
struct struct-name variable name;
```

3. Access the structure elements:

```
variablename.elementname;
```

Example-Structure Program

Input 10 students Rollno, Name and 3 subject marks and display the same with result.

```
#include <iostream.h>
#include <conio.h>
struct student
{
        long int rollno;
        char na[20];
        int t,e,m;
};
struct student stu[10];
void main()
{
        int i;
        cout << "Enter 10 student records";
        for(i = 0; i < 10; i++)
        {
                cout << "Enter " << i+1 << " student record";
                cout << "Enter Rollno";
                cin >> stu[i].rollno;
                cout << "Enter Name";
                cin >> stu[i].na;
                cout << "Enter 3 subject marks";
                cin >> stu[i].t,stu[i].e,stu[i].m;
        }
        clrscr();
        cout << "\n----------------------------------------“;
        cout << "\nRollno\tName\tTamil\tEnglish\tMaths\tResult";
        cout << "\n----------------------------------------“;
```

```
        for(i =0; i < 10;i++)
        {
                cout << "\n" << stu[i].rollno << "\t" <<      stu[i].na <<     "\t" <<   stu[i].t << "\t" <<
                stu[i].e          << "\t" << stu[i].m << "\t";
                if (stu[i].t < 35 || stu[i].e < 35 || stu[i].m < 35)
                cout << "fail";
                else
                cout << "Pass";
        }
        cout << "\n----------------------------------------“;
        getch();
}
```

output

```
-----------------------------------------------------
rollno  name   tamil   english maths  result
-----------------------------------------------------
1001    ganesh 30      50      60      fail
1002    siva   56      60      90      pass
-----------------------------------------------------
```

Structures within Structures

A structure can be included within another structure.

For example:

```
struct date
{
        int day;
        int month;
        int year;
};
struct stud
{
        long int rollno;
        char name[20];
        struct stud dob;
};
```

To Declare Variable:

```
        struct stud stu;
```

To access members:

```
        stu.rollno
        stu.name
        stu.dob.day
        stu.dob.month
        stu.dob.year
```

9. UNION

- Single piece of memory that is shared by two or more variables.
- Variables that share the memory location may differ in type.
- However only one variable can be used by the memory location at a time. Like structure, unions are user defined data types.
- The size of the union is the largest member of the union.

Union Declaration

The union can be declared in the same way as the structure.

```
union union-tag
{
        member1;
        member2;
          .
};
```

Example:

```
#include<stdio.h>
union a
{
        int i;
        char ch[2];
};
void main()
{
        union a key;
        clrscr();
        key.i=512;
        printf("key.i=%d\n", key.i);
        printf("key.ch[0]=%d\n", key.ch[0]);
        printf("key.ch[1]=%d\n", key.ch[1]);
        getch();
}
```

Output

```
key.i=512
key.ch[0]=0
key.ch[1]=2
```

10. Stack Vs Heap Memory

Discussion – 1

Memory stacks are dynamic groupings of memory growing and shrinking as each program allocates and deal locates memory.

The heap is an area of unused memory managed by your Operating System.

Discussion - 2

Both are used for memory management. The stack is used to store your code, and runtime values: parameters return address, links, return value and local variables.

Heaps are used to contain all your dynamically allocated variables. A heap maintains a free list (list of all the free space available) and as and when you allocate and deallocate memory it updates this free list.

Discussion – 3

What is heap and stack?

The **stack** is a place in the computer memory where all the variables that are declared and initialized *before* runtime are stored. The **heap** is the section of computer memory where all the variables created or initialized *at* runtime are stored.

What are the memory segments?

The distinction between stack and heap relates to programming. When you look at your computer memory, it is organized into three segments

- Text (code) Segment
- Stack Segment
- Heap Segment

Text Segment: is where the compiled code of the program itself resides. Compiled code (for windows exe file) is the machine code, the computer representation of the programming instructions. This include all user defined as well as system functions.

STACK AND HEAP

The other two segments (stack and heap) in the memory are used to store data. The stack is used to store automatic variables within functions. Data is stored in stack using Last in Fast Method (LIFO). This means that storage in the memory is allocated and deallocated at only one end of the memory called the top of the stack. On the other hand, heap is an area of memory used for dynamic allocation of memory. The size of allocation block is not known until run time. The stack is much faster than the heap but also smaller and more expensive.

Heap and Stack from Program Perspective

Most object-oriented languages have some defined structure, and some come with so-called main() function. When a program begins running, the system calls the function main() which marks the entry point of the program. For example every C, C++, or C# program must have one function named main(). No other function in the program can be called main(). Before we start explaining, let's take a look at the following example:

```
Int x;  /* statick storage */
Void main()
{
        Int y; /* dynamic stack storage  */
        Char str; /* dynamic stack storage  */
        Str = malloc(50);  /* allocates 50 bytes of dynamic heap storage */
        Size = calcsize(10);  /* dynamic heap storage  */
}
```

When a program begins executing in the main() function, *all variables declared within main() will be stored on the stack*. If the main() function calls another function in the program, for example calcSize(), additional storage will be allocated for the variables in calcSize(). This storage will be allocated in the **heap** memory segment.

Notice that the parameters passed by main() to calcSize() are also stored on the stack. If the calcSize() function calls to any additional functions, more space would be allocated at the heap again.

When the calcSize() function returns the value, the space for its local variables at heap is then deallocated and heap clears to be available for other functions.

The memory allocated in the heap area is used and reused during program execution.

It should be noted that memory allocated in heap will contain garbage values left over from previous usage.

Memory space for **objects** is always allocated in heap. Objects are placed on the heap. **Built-in datatypes** like int, double, float and parameters to methods are allocated on the stack. Even though objects are held on heap, references to them are also variables and they are placed on stack.

The stack segment provides more stable storage of data for a program. The memory allocated in the stack remains in existence for the duration of a program. This is good for global and static variables. Therefore, global variables and static variables are allocated on the stack.

Why is stack and heap important?

When a program is loaded into memory, it takes some memory management to organize the process. If memory management was not present in your computer memory, programs would clash with each other leaving the computer non-functional.

11. STORAGE CLASS

- Storage class defines both scope and lifetime of a variable in a program. The default storage class is auto. (equal to local)
- There are two kinds of locations in a computer where the values are kept. One is memory (RAM) and another one is CPU register.
- A variable storage class tells us
  - where the variable would be stored
  - What will be the initial value, if it is not assigned?
  - What is the scope of the variable
  - What is the life of the variable

There are four type of storage

1. Auto
2. Register
3. Static
4. External

1. Auto:

Storage - Memory
Default value - Garbage value
Scope - Local to the block in which the variable is defined.
Life - Till the control remains within the block in which it is defined.

Example:

```
void increment()
{
        auto int i=1;
        printf("%d\n", i);
        i=i+1;
};
void main()
{
        increment();
        increment();
        increment();
        getch();
}
```

Output:

```
1   1   1
```

2.Register Storage Class:

Storage - CPU register
Default value - garbage
Scope - Local to the block
Life - Till the control remains within block Value stored in CPU register can always be accessed faster than the one which is stored in memory.

Example:

```
void main()
{
        register int i;
        for(i=0;i<=10;i++)
        printf("%d\n",i);
}
```

3. Static Storage Class:

Storage - Memory
Default value - Zero
Scope - Local to the block in which the variable is defined
Life - Value of the variable remains between different function class. A static variable declared inside a function, remain its value after the function termination.

Example:

```
void increment()
{
        static int i=1;
        printf("%d\n", i);
        i=i+1;
};
void main()
{
        increment();
        increment();
        increment();
        getch();
}
```

Output:

```
        1   2   3
```

4. External Storage Class:

Storage - Memory
Default value - Zero
Scope - allowed
Life - as long as the program execution

External variable are declared above all functions. This variable is available and can access from all function.

12. FUNCTION / PROCEDURE

- A function is a small segment of program that carries out some specific task when called and return the execution back to the calling program.
- The most important of the 'C' program is the main() function, whenever a C program is executed, execution starts from the main function.
- Some of the other functions are printf(), scanf(), getch() etc.,
- When large programs are coded, a part of the program code may have to be repeatedly written.
- The repeated program segment that can be placed separately from the main program is called a function.
- This function can be called from different part of the main program,whereever required.

Format
Return Type Function Name (Arg List)
{
    variable decleration;
    executable statements;
    return (value);
}

Return Data Types
int, long int, float, double, char - for functions
void - for procedure
if void, then no return(value) statement is required inside the function.

Argument List

Datatype v1, Datatype v2,.....

Example one integer input
    (int a)
two integer input
    (int a, int b)
one integer, one float
    (int a, float b)
no arquements
    () or (void)

Argument Default Value

Example -for default value

```
void test(int a = 10, int b = 20)
(
    cout << "a = " << a;
    cout << "b = " << b;
}
void main()
```

```
{
        test();
        test(5);
        test(1,2);
}
```

Output:

```
        A = 10 b = 20
        A = 5 b = 20
        A = 1 b = 2
```

Default Return type in c is int

```
        main() =  int main()
int main()
{
        return (0);
}
void main()
{}
```

Using Arguments and Return Types Functions Are Classified as Four Types

1. without argument, without return type
2. with argument, Without return type
3. without argument, with return type
4. with argument, with return type

1 without Argument, without Return type

Example :

```
void line()
{
        int i;
        for (i = 1; i < 80; i++)
        cout << "-";
        cout << "\n";
}
void main()
{
        line();
        cout << "welcome to IWS";
        line();
}
```

```
--------------------------------
welcome to IWS
--------------------------------
```

2. with Argument, Without Return type

Example -1:

```
void line(int a)
{
        int i;
        for (i = 1; i < a; i++)
        cout << "-";
        cout << "\n";
}
void main()
{
        line(50);
        cout << "welcome to IWS";
        line(70);
}
```

```
----------------------
welcome to iws
--------------------------------
```

2. with Argument, Without Return type

Example - 2:

```
void table(int n)
{
        int i,j;
        for(i = 1; i <= 20;i++)
        {
                j = i * n;
                cout << "\n" << i << " x " << n << " = "
<< j;
        }
}
void main()
{
        int a;
        cout << "Enter a value";
        cin >> a;
        table(a);
}
```

3. without Argument with Return type

Example :

```
int big()
{
        int a,b;
        cout << "Enter 2 values";
        cin >> a >> b;
        if (a > b)
                return (a);
        else
                return (b);
}
void main()
{
        int c;
        c = big();
        cout << "biggest value = "
<< c;
}
```

4. with Argument, With Return type

Example – 1:

```
int fact(int n)
{
        int i=1,j=1;
        while (i <= n)
        {
                j = j *i;
                i++;
        }
        return (j);
}
void main()
{
        int a,b;
        cout << "Enter a value";
        cin >> a;
        b = fact(a);
        cout << a << " factorial value = " << b;
}
```

4. with Argument, With Return type

Example - 2:

```
int sumofdigit(int n)
{
        int j = 0;
        while (n > 0)
        {
                j = j + (n %
10);
                n = n/ 10;
        }
        return (j);
}
void main()
{
        int a,b;
        cout << "Enter a
value";
        cin >> a;
        b = sumofdigit(a);
```

	cout << "sum of digit value " << b; }

II. Passing Single Dimensional Array in Parameter

- An Array can be passed as an argument in a function.
- When passing an array, the array name without square bracket should be given in the actual parameter.
- In the formal parameter, the array name should be given with [] and another parameter gets the number of elements in the array.
- Array passed as reference hence the value changes in the formal parameter variable reflect to the main program. No return value is required in the sub function.

Example:

```
#include<stdio.h>
void sort(int no[ ], int m)
{
        int i, j, temp;
        for(i=0; i<m-1; i++)
        {
                for(j=i+1; j<m; j++)
                {
                        if(no[i]>no[j])
                        {
                                temp=no[i];
                                no[i]=no[j];
                                no[j]=temp;
                        }
                }
        }
}
```

```
void main()
{
        int i, num[5];
        printf("Enter 5 values");
        for(i=0; i<5; i++)
        scanf("%d", &num[i]);
        sort(num,5);
        printf("\n given values in    ascending order:");
        for(i=0; i<5; i++)
        printf("\t%d", num[i]);
        getch();
}
```

III. Passing a Multi Dimensional Array as Parameters:

- In actual parameter pass the array name with row index and column index.
- In formal parameter row index should be empty and column index should be given.
- Parameters pass by reference only, hence the value change in the function should reference in the main program.

Example : Refer Example 3.1

IV.Recursion Function

A function call itself

Example : Refer Example 4.1

Example 3.1:

```
#include<stdio.h>
#include<conio.h>
void trans (int a[][10],int r1,int c1)
{
        int i,j;
        for (j=0;j<c1;j++)
        {
                for(i=0;i<r1;i++)
                {
                        printf("%d\t",a[i][j]);
                }
                printf("\n");
        }
}
```

```
void main()
{
        int a[10][10],r1,c1,r2,c2;
        printf("Enter the Row,Col value for
Matrix");
        scanf("%d%d",&r1,&c1);
        printf("\n enter value for matrix a");
        for(r2=0;r2<r1;r2++)
        {
                for(c2=0;c2<c1;c2++)
                {
                        scanf("%d",&a[r2][c2]);
                }
        }
        trans(a,r1,c1);
        getch();
}
```

Example 4.1:

```
int fact(int n)
{
        int j;
        if (n == 1)
                return (1);
        else
                j = n * fact(n-1);
        return (j);
}
```

```
void main()
{
        int a,b;
        cout << "Enter a value";
        cin >> a;
        b = fact(a);
        cout << a << " factorial value = " << b;
}
```

V.Arguments Passing by Value

If any changes in the formal parameter will not affect the actual parameter

Example :

```
void swap(int a, int b)  // formal parameters
{
        int c;
        c = a;
         a = b;
         b = c;
        cout << "\ninside swap a=" << a << " b=" << b;
}
```

```
void main()
{
        int a = 10, b = 20;
        cout << "\nBefore swap a=" << a << " b=" << b;
        swap(a,b);   // actual parameters
        cout << "\nAfter swap  a=" << a << " b=" << b;
}
```

OUTPUT

before swap a= 10 b = 20
inside swap a = 20 b = 10
after swap a = 10 b = 20

VI.Passing Argument By Reference

Change the value of formal parameter will also be done in the actual parameter.

c++ program

```
void swap (int &a, int &b)
{
        int c;
        c = a;
        a = b;
        b = c;
        cout << "inside swap a=" << a << " b =" << b;
}
```

```
void main()
{
        int a= 10,b= 20;
        cout << "before swap a= " << a << " b=" << b;
        swap(a, b);
        cout << "after swap a= " << a << " b= " << b;
}
```

OUTPUT:

before swap a= 10 b = 20
inside swap a = 20 b = 10
after swap a = 20 b = 10

VII.Library Functions

- Library functions are pre defined functions that C program offers to user.
- Library functions are grouped in header files.
- To invoke C Library function, corresponding header file should be included.

Header File	**Function**	**Example**
studlib.h	atof() atoi() atoll() itoa()	string to float float avg=atoi(“3303); string to integer int tot=atoi(“100”); string to long int long int rollno=ctol(“547879”); integer to string
string.h	strcat() strlwr() strrev() strcmp() strlen() strcpy() strset()	strcat() strlwr() strrev() strcmp() return number of variables strlen(“raja”)=4 copy one string data into another variable strcpy(s1,s2) s1=s2=ram; replace all the character in a string with characters.
ctype.h	isascii() toupper() isupper() tolower() islower() isalpha() isdigit()	find ascii value of alphabets convert uppercase check uppercase convert lowercase check lowercase check alphabet or numeric check for digit

VIII. Function Overloading (C++ Only Support)

Two or more function with same name, but differs in return type or argument list.

Example :

```
void big(int a, int b)
{
        if (a > b)
         cout << " big = " << a;
        else
         cout << " big = " << b;
}
void big(int a, int b, int c)
{
        if (a>b && a > c)
        cout << "big = " << a;
        else if (b > c)
        cout << "big = " << b;
        else
        cout << "big = " << c;
}
```

```
void main()
{
        int a,b,c;
        cout << "Enter 2 values";
        cin >> a >> b;
        big(a,b);
        cout << "Enter 3 values";
        cin >> a >> b >> c;
        big (a,b,c);
}
```

13. POINTERS

Pointer is a variable use to store address of a memory location.

Define A Pointer Variable

datatype *variablename;

Store Address to Pointer

1. Another variable Address
   pointervariable = &variable
2. Dynamic Memory allocation using malloc
   pointervariable = malloc(n)
   where n – size of bytes
3. Dynamic Memory Allocation using calloc
   pointervariable = calloc(m,n)

   where m – number of variables
   n – size of each variables

Note: Difference Between malloc, calloc and new

malloc allocates uninitialized memory. The allocated memory has to be released with **free**.

calloc is like malloc but initializes the allocated memory with a constant (0). It needs to be freed with free.

new initializes the allocated memory by calling the constructor (if it's an object). Memory allocated with new should be released with delete (which in turn calls the destructor). It does not need you to manually specify the size you need and cast it to the appropriate type. Thus, it's more modern and less prone to errors

- **calloc(m,n) is equal to**
- **p = m*malloc(n) and**
- **memset(p,0,m*n)**

Read Data Using Pointer Variable

*pointervariable

Operators Are Used With Pointers

& - address off (unary operator)
* - value at (dereference operator)

Operators Are Used With Pointers

Example -1:

```
#include  <stdlib.h>
void main()
{
        int a,*b,*c;
        a = 50;
        b = &a;
        c = (int *)malloc(2);
        *c = 100;
        cout << "\nvalue of a = " << a;
        cout << "\nAddress of a = " << &a;
        cout << "\nValue of b = " << b;
        cout << "\nvalue at b = " << *b;
        cout << "\nAddress of b = " << &b;
        cout << "\n value of c = " << c;
        cout << "\n value at c = " << *c;
        cout << "\n Address of c = " << &c;
        free(b);
        free(c);
}
```

II.Pointer to One Dimentional Array

Example – 2:

```
#include  <stdlib.h>
void main()
{
        int a[5],*b,i,j;
        b = a;
        cout << "Enter 5 values";
        for(i = 0;i<5;i++)
        cin >> a[i];
        cout << "\n Display given values using pointer variable";
        for(i = 0;i<5;i++)
        {
                cout << "\n" << *b;
                b++;
        }
        free(b);
}
```

III.Pointer to Two Dimensional Arrays

To read/write Data of Two Dimensional Array using Pointer

*(variable + i)	Base Pointer of the ith row
*(variable + i) + j	Pointer of the ith row and jth column
* (*(variable + row) + col)	Value of the the ith row and jth column

Example – 3:

```
#include  <stdlib.h>
void main()
{
        int a[3][2],(*b)[2],i,j;
        cout << "Enter A matrix value";
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                cin >> a[i][j];
        }
        b = a;
        cout << "Display given data using pointer";
        for(i=0;i<3;i++)
        {
                cout << "\n";
                for(j=0;j<2;j++)
                cout << "\t" << *(*((b+i)+j));
        }
        free(b);
}
```

Pointer to Two Dimentional Arrays

Step - 1:

To declare pointer to two dimensional arrays

int (*pt)[10];

***pt is the pointer to an array of 10 integers;**

e.g:

```
int arr[10];
int (*pt)[10];
pt = arr;
```

note sizeof(*pt) =: 20 bytes

step - 2:

int *pt[10];

here pt is the pointer array. That is we can store 10 integer variable addresses in pt.

step - 3:

```
int arr[5][10];
int (*pt)[10];
pt = &arr[0];
```

Now pt can access 10 elements of the first row

pt + 1;

Increment 20 byes to access to the next row.

IV.Pointer To Structure

To access members using structure pointer

pointer->member;

Example - 4:

```
#include <stdlib.h>
struct student
{
        long int rollno;
        char na[20];
        int t,e,m;
};
```

```
void main()
{
        int i,j,k;
        struct student *pt,*pt1;
        cout << "How many records going to input";
        cin >> i;
        pt = (struct student *)malloc(i * sizeof (struct student));
        pt1 = pt;
        cout << "Enter " << i << " Records";
        for(j=0;j< i;j++)
        {
                cout << "Enter Rollno";
                cin >> pt->rollno;
                cout << "Enter name";
                cin >> pt->na;
                cout << "Enter 3 subject marks";
                cin >> pt->t >> pt->e >> pt->m;
                pt++;
        }
        clrscr();
        cout << "\n----------------------------";
        cout <<  "\nRollno \tName\tTamil\tEnglish\tMaths\tResult";
        cout << "\n----------------------------";
        pt =pt1;
        for(j=0;j<i;j++)
        {
                cout << "\n" << pt->rollno << pt->na <<
                pt->t << pt->e << pt->m;
                if (pt->t < 35 || pt->e < 35 || pt->M < 35)
                cout << "fail";
                else
                cout << "pass";
                pt++;
        }
        cout << "\n-----------------------------";
        free(pt);
        free(pt1);
}
```

V.Pointer to String

```
        char *na;
        cin >> na;
        cout << "Given Name " << na;
```

Example :

input- raja
raja store as raja\0

the first address of r = 1001
then
na = 1001;

char *na[10]; store 10 string address

Example – 5:

Input 5 names and display given names in ascending order.

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
void main()
{
        int i,j,k;
         char *na[5],na1[20];
         cout << "Enter 5 names";
        for(i = 0;i < 5; i++)
        {
        cin >> na1;
        na[i] = (char *)malloc(strlen(na1)+1);
        strcpy(na[i],na1);
        for(i = 0;i < 4; i++)
        {
                for(j = i + 1; j < 5; j++)
                {
                        if (strcmp(na[i],na[j]) > 0)
                        {
                                strcpy(na1,na[i]);
                                strcpy(na[i],na[j]);
                                strcpy(na[j],na1);
                        }
                }
         }
        cout << "\n Given Names in ascending order";
        for (i=0;i<5;i++)
        cout << "\t" << *na[i];
        getch();
        free(na);
        free(na1);
}
```

Example – 6:

String Copy Using Pointer

```
char *my_strcpy(char dest[], char source[])
{
        int i = 0;
        while (source[i] != '\0')
        {
                dest[i] = source[i];
                i++;
        }
        dest[i] = '\0';
        return dest;
}
```

Note : here
*(dest + i) = *(source+i)
is equal to dest[i] = source[i]

VI. Pointer to Function

datatype (*function)(arguement);

Example : Refer Example 10

Example – 9:

String Concatenates using pointer

```
void strappnd(char *dest, char *src)
{
        /* appends one string to another */
         while(*dest != '\0')
         {
               dest++;
         }
        while (*src != '\0')
                 *dest++ = *src++;
        *dest = '\0';
}
```

Example – 10:

```
#include <iostream.h>
#include<conio.h>
int big(int a, int b)
{
        int c;
        if (a > b)
                c = a;
        else
                c = b;
        return(c);
}
void main()
{
        int (*f1)(int,int);
        int d;
        f1 = &big;
        d = f1(10,20);
        cout << "biggest value = " << d;
        free(f1);
}
```

Pointer Summary

Declaration	Description
Int *ptr	Ptr is a pointer to an integer quantity
Char *ptr[8]	Ptr is an one-dimensional array of pointers to single characters or string which is equivalent to a double dimensional array of characters
Int (*ptr)[8]	Ptr is a pointer to a 8 element integer array
Int *func(void)	Func is a function that returns a pointer to an integer quantity
Int *func(int *)	Func is a function that accepts an integer argument and returns a pointer to an integer
Int (*fptr)(int x)	Fptr is a pointer to a function that accepts an integer argument and return an integer
Int (*funct(char *x))	Func is a function that accepts an argument which a pointer to a character and returns a pointer to an integer

Int (*func(chart *x))[8]	Func is a function that accepts an argument which is a pointer to a character and returns a point to a 8 – element integer array
Int func(char (*x)[]);	Func is a function that accepts an argument which is a pointer to a character array and returns an integer quantity
Int func(char *x[]);	Func is a function that accepts an argument which is an array of pointers to characters and returns an integer quantity
Int * (*fptr)(char (*x)[])	Fptr is a pointer to a function that accepts an agreement which is a pointer to an character array and returns a pointer to an integer quantity
Int * (*fptr[8](char *arr)	Fptr is an 8 element array of pointers to function, each function accepts an argument which is a character, and returns a pointer to an integer quantity.

14. FILE OPERATION

1. Create a File Pointer

```
FILE *pt;
```

2. Open a file to the pointer

```
pt = fopen("filename","mode");

mode - > "w", "a", "r", "w+","a+','r+",
          “wb”,”rb”,”ab”,”wb+”,”rb+”,”ab+”
filename -> userdefined
```

3. Read or Write data to a file

```
Write - char by char putc(ch,pt);
read   - char by char ch = getc(pt);
```

4.Word By Word

```
write - fprintf(pt,"format",variablelist);
read  - fscanf(pt,"format",variablelist);
```

5. String

```
write - fputs(string,pt);
read   - fgets(string,length,pt)
```

6. Record By Record

```
write -  fwrite(&stru,sizeof(stru),1,pt);
read  - fread(&stru,sizeof(stru),1,pt);
```

7.Close A File

```
fclose(pointer);
```

Example - 1:

Copy the data from prog1.c to prog2.c

```
#include <stdio.h>
#include<conio.h>
void main()
{
        FILE *fp1,*fp2;
        int c;
        fp1 = fopen("prog1.c","r");
        fp2 = fopen("prog2.c","w");
        if (fp1 == NULL)
        {
                printf("Cannot open prog1.c for reading");
                exit();
        }
        else if(fp2 == NULL)
        {
                printf("Cannot open prog2.c for reading");
                exit();
        }
        else
        {
                 c= getc(fp1);
                while(c != EOF)
                {
                         putc(c,fp2);
                        c = getc(fp1);
                }
                fclose(fp1);
                fclose(fp2);
                printf("files successfully copied..");
        }
        free(fp1);
        free(fp2);
}
```

Example – 2:

Copy the data using command line arguments

```
#include <stdio.h>
#include <conio.h>
void main(int argc, char *argv[])
{
        FILE *fp1,*fp2;
        int c;
        if (argc != 3)
        {
                printf("Error in arguments");
                exit();
        }
        fp1 = fopen(argv[1],"r");
        fp2 = fopen(argv[2],"w");
        if (fp1 == NULL)
        {
                printf("Cannot open prog1.c for reading");
                exit();
        }
        else if(fp2 == NULL)
        {
                 printf("Cannot open prog2.c for
reading");
                exit();
        }
        else
        {
                c= getc(fp1);
                while(c != EOF)
                {
                        putc(c,fp2);
                        c = getc(fp1);
                }
                fclose(fp1);
                fclose(fp2);
                printf("files successfully copied..");
        }
        free(fp1);
        free(fp2);
}
```

Example – 3:

Program to perform file operations using structure

Write records in Binary Format

```
#include <stdio.h>
void main()
{
        FILE *fp;
        char another = 'Y';
        struct emp
        {
                char name[30];
                int age;
                float bas;
        }e;
        fp = fopen("c:\\emp.dat","wb");
         if (fp == NULL)
        {
                puts("Cannot Open File");
                exit();
        }
        while(another == 'Y')
        {
                printf("\n enter name, age and basic");
                scanf("%s %d %f",e.name,&e.age,&e.bas);
                fwrite(&e,sizeof(e),1,fp);
                printf("Another Record (Y/N)");
                fflush(stdin);
                another = getche();
        }
        fclose(fp);
        free(fp);
}
```

Example – 4:

Read Records in Binary Format

```
#include <stdio.h>
#include <conio.h>
void main()
{
        FILE *fp;
        char another = 'Y';
        struct emp
        {
                char name[30];
                int age;
                float bas;
        }e;
        fp = fopen("emp.dat","rb");
        if (fp == NULL)
        {
                puts("Cannot Open File");
                exit();
        }
        while(fread(&e,sizeof(e),1,fp) == 1)
        printf("%s %d %f",e.name,e.age,e.bas);
        getch();
        fclose(fp);
        free(fp);
}
```

Random Access File Handling Techniques

To access the file in a random fashion i.e. a particular area of the file to be processed use the following function.

int fseek(FILE *pt, long offset, int whence);

Where pt is the file pointer, offset is the number of bytes to be moved, offset value will be positive or negative, whence indicates the starting point.

The whence can be one of three values namely.

SEEK_BEG -- to indicate the beginning of the file
SEEK_CUR -- to indicate the current position of the file.
SEEK_END -- to indicate the end of the file.

Example

fseek(fp,10L, 0) or fseek(fp, 10L, SEEK_BEG)

File System Functions

ftell() function:

- This function returns the current position of the file pointer in the form of a long number.
- The number is the relative offset of the current file pointer position in terms of bytes.

long int ftell(FILE *pt);

rename() function

- This function renames the file in argument.

int rename(oldfilename,newfilename);

rewind() function

- This function resets the pointer to the beginning of the file.

void rewind(FILE *pt);

remove() function:

- This function deletes a file.

remove(filename);

15. GRAPHICS AND MOUSE OPERATIONS

include <graphics.h>

c - supports two modes

1. Text 2. Graphics

- Text mode support 25 rows and each row 80 columns.
- Graphics mode has points x- 640 y - 480

Loading the header file of C graphics.h

The C graphics.h is the header file which should be included to initialize your computer to start using graphics methods and also to initialize the monitor. These kinds of statements before the main program are called preprocessor dierctives.Its very simple as coded below.

Initializing 'Graphics Driver' and 'Graphics Mode' from Borland C graphics library

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int gdriver=DETECT,gmode;
        initgraph(&gdriver,&gmode," ");
}
```

Initializing Graphics Form C graphics library

This initialization is done inside the main program. The method **initgraph()** is used for this purpose there are three parameters for this , the first two are of integer type and the next is the path which you can leave blank in quotes to take the default path.The integer parameters are initialized in this method for all the C graphics programs so you just need to memorize it and apply it for all C graphics programming from now on.

Some Common C Graphics Programming Methods

Well, these are some of the most popular C Graphics Methods used in while graphics programming soon after you learn these you can think of developing small games , puzzles and so on. I'll just give only a small overview on each methods.x,y,x1,y1,x2,y2 etc are all variables you'll have to substitute with numeric values to get working programs. Further clear-cut reference can be obtained from any of the C graphics book available from Amazon or your local library (These books are pretty old).

Cleardevice()

Clears all previous graphical outputs generated by the previous programs. It's a good practice to include this method at the starting of each program.

gotoxy()

This will initialize the graphics cusor to the specified co-ordiante.In C gotoxy function is used very frequently to locate the cursor at different locations whenever as necessary.

putpixex()

It will color the pixel specified by the co-ordinates.

outtextxy()

This method is used to display a text in any position on the screen. The numeric coordinates are substituted for x and y.

rectangle()

Draws a rectangle according to the given parameter x and y are the top-left corner co-ordinates.

circle()

Draws a circle with x,y as the center .

line()

Draws a line as per the given co-ordinates.

moveto()

Cursor is moved from the current location to the specified location dx,dy.These parameters can also be used as incremental values.

lineto()

Draws a line from its current location to the co-ordinate(x,y)

ellipse()

Draws the ellipse with the specified angles and coordinates.

drawpoly()

Draws a polygon with (num_of_points +1) edges.The array 'points'

int points[ ]=(x1,y1,x2,y2,x3,y3...)

settextstyle()

The fonts available are :**TRIPLEX_FONT, SMALL_FONT, SANS_SERIE_FONT, GOTHIC_FONT**

The direction can be changed as **HORIZ_DIR** or **VERT_DIR**

The charecter size increases from **1** to **10**

setfillstyle()

The fill styles avaliable are SOLID_FILL, LINE_FILL, HATCH_FILL, SLASH_FILL etc.

setcolor()

Sets the color

floodfill()

The function fills the enclosed area about x,y with the highest value of the color returned by the **getmaxcolor()**.

setviewport()

This marks a rectangle starting from (10,10) having widht 630 pixels height 470 pixels.The integer specifies this identified area may be clipped. **clearviewport()** reverse this function.

itoa()

Converts the integer x1into an alphabet, and puts it in st1, which is an array to hold 10 characters.10 is the buffer size.

delay()

Cause a pause in execution of the program 1000ms = 1 second

closegraph()

Terminates all graphics operations and revert the hardware back to the normal mode.

Example :

```
1. cleardevice()
2. gotoxy(x,y)
3. putpixel(x,y,WHITE)
4. outtextxy(x,y,"HELLO")
5. rectangle(x,y,int x_width,int y_height)
6. circle(x,y,radius)
7. line(x1,y1,x2,y2)
8. moveto(dx,dy)
9. lineto(x,y)
10. ellipse(x-centre,y-center,starting_angle,ending_angle,x_radius,y_radius)
11. drawpoly(num_of_points + 1, points)
12. settextstyle(DEFAULT_FONT,HORIZ_DIR,1)
13. setfillstyle(SOLID_FILL,RED)
14. setcolor(CYAN)
15. floodfill(x,y,getmaxcolor())
16. setviewport(int left,int top,int right,int bottom, int clip)
17. setviewport(10,10,630,470,1)
18. itoa(x1,st1,10)
19. delay(100)
20. closegraph()
```

Example Program:

Write a Cgraphics program, that input 2 values and display the result using basic arithmetic operations (add, sub, mul and div).

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
union REGS in, out;

void display();
void detectmouse ()
{
        in.x.ax = 0;
        int86 (0X33,&in,&out);
 }
 void showmouse()
 {
        in.x.ax = 1;
        int86(0X33,&in,&out);
 }
 void showmousegraphics()
 {
        in.x.ax = 1;
        int86(0X33,&in,&out);
}
  void hidemouse()
 {
        in.x.ax = 2;
        int86(0X33,&in,&out);
 }
 void detect()
 {
        int x,y;
        int x1,y1;
        int i;
        int j;
        int esc1=0;
        char ch[20]="";
        char ch1[20]="";
        char ch2[20]="";
        int chrcount=0;
        int chrcount1=0;
        i = 1;
        while (i != 27)
        {
                while (!kbhit())
                {
```

```
in.x.ax = 3;
int86(0X33,&in,&out);
if (out.x.bx == 1)
{
        x = out.x.cx;
        y = out.x.dx;
        if ((x >= 350 && x <= 450) && (y >= 130 && y <= 160))
        {
                x = 350;
                y = 132;
                setcolor(LIGHTGRAY);
                rectangle(350,130,352,160);
                setfillstyle(SOLID_FILL,LIGHTGRAY);
                floodfill(352,140,LIGHTGRAY);
        }
        else if ((x >= 350 && x <= 450) && (y >= 180 && y <= 210))
        {
                x = 350;
                y = 182;
                setcolor(LIGHTGRAY);
                rectangle(350,180,352,210);
                setfillstyle(SOLID_FILL,LIGHTGRAY);
                floodfill(352,190,LIGHTGRAY);
        }
        else
        {
                for(x1 = 350;x1 <=450;x1++)
                for(y1 = 230;y1<=260;y1++)
                putpixel(x1,y1,WHITE);
                if ((x >= 180 && x <= 220) && (y >= 300 && y <= 330))
                {
                        x = 350;
                        y = 232;
                        setcolor(LIGHTGRAY);
                        j= atoi(ch) + atoi(ch1);
                        itoa(j,ch2,10);
                        outtextxy(x,y,ch2);
                }
                else if ((x >= 230 && x <= 270) && (y >= 300 && y <= 330))
                {
                        x = 350;
                        y = 232;
                        setcolor(LIGHTGRAY);
                        j= atoi(ch) - atoi(ch1);
                        itoa(j,ch2,10);
                        outtextxy(x,y,ch2);
                }
                else if ((x >= 280 && x <= 320) && (y >= 300 && y <= 330))
                {
                        x = 350;
```

```
                                        y = 232;
                                        setcolor(LIGHTGRAY);
                                        j= atoi(ch) * atoi(ch1);
                                        itoa(j,ch2,10);
                                        outtextxy(x,y,ch2);
                                }
                                else if ((x >= 330 && x <= 370) && (y >= 300 && y <= 330))
                                {
                                        x = 350;
                                        y = 232;
                                        setcolor(LIGHTGRAY);
                                        j= atoi(ch) / atoi(ch1);
                                        itoa(j,ch2,10);
                                        outtextxy(x,y,ch2);
                                }
                                else if ((x >= 380 && x <= 420) && (y >= 300 && y <= 330))
                                {
                                        esc1 = 1;
                                        break;
                                }
                        }
                }
                out.x.ax = 0;
                delay(100);
        }
        if (esc1 == 0)
        i = getch();
        else
        i = 27;
        if (i >=41 && i <= 60)
        {
                if (y < 180)
                {
                        ch[chrcount] = (char)i;
                        ch[++chrcount] = '\n';
                        settextstyle(1,0,1);
                        outtextxy(x,y,ch);
                }
                else
                {
                         ch1[chrcount1] = (char)i;
                         ch1[++chrcount1] = '\n';
                         settextstyle(1,0,1);
                         outtextxy(x,y,ch1);
                 }
        }
    }
}
void display()
{
```

```
int gd = DETECT, gm;
int t,i;
long int k,j;
initgraph(&gd,&gm,"");
setcolor(LIGHTGRAY);
rectangle(0,0,getmaxx(),getmaxy());
setfillstyle(SOLID_FILL,LIGHTGRAY);
floodfill(getmaxx()/2,getmaxy()/2,LIGHTGRAY);
setcolor(DARKGRAY);
rectangle(50,50,getmaxx()-50,getmaxy()-50);
setfillstyle(SOLID_FILL,DARKGRAY);
floodfill(getmaxx()/2,getmaxy()/2,DARKGRAY);
setcolor(RED);
outtextxy(200,150,"Enter_I_value");
setcolor(WHITE);
rectangle(350,130,450,160);
setfillstyle(SOLID_FILL,WHITE);
floodfill(370,150,WHITE);
setcolor(RED);
outtextxy(200,200,"Enter_iI_value");
setcolor(WHITE);
rectangle(350,180,450,210);
setfillstyle(SOLID_FILL,WHITE);
floodfill(400,200,WHITE);
setcolor(RED);
outtextxy(200,250,"Result");
setcolor(WHITE);
rectangle(350,230,450,260);
setfillstyle(SOLID_FILL,WHITE);
floodfill(400,240,WHITE);
setcolor(RED);
rectangle(180,300,220,330);
setfillstyle(SOLID_FILL,RED);
floodfill(200,320,RED);
setcolor(LIGHTGRAY);
rectangle(184,304,216,326);
setfillstyle(SOLID_FILL,LIGHTGRAY);
floodfill(200,310,LIGHTGRAY);
setcolor(RED);
outtextxy(190,310,"Add");
setcolor(RED);
rectangle(230,300,270,330);
setfillstyle(SOLID_FILL,RED);
floodfill(250,320,RED);
setcolor(LIGHTGRAY);
rectangle(234,304,264,326);
setfillstyle(SOLID_FILL,LIGHTGRAY);
floodfill(240,310,LIGHTGRAY);
setcolor(RED);
outtextxy(240,310,"Sub");
```

```
        setcolor(RED);
        rectangle(280,300,320,330);
        setfillstyle(SOLID_FILL,RED);
        floodfill(300,320,RED);
        setcolor(LIGHTGRAY);
        rectangle(284,304,316,326);
        setfillstyle(SOLID_FILL,LIGHTGRAY);
        floodfill(300,310,LIGHTGRAY);
        setcolor(RED);
        outtextxy(290,310,"Mul");
        setcolor(RED);
        rectangle(330,300,370,330);
        setfillstyle(SOLID_FILL,RED);
        floodfill(350,320,RED);
        setcolor(LIGHTGRAY);
        rectangle(334,304,366,326);
        setfillstyle(SOLID_FILL,LIGHTGRAY);
        floodfill(340,310,LIGHTGRAY);
        setcolor(RED);
        outtextxy(340,310,"Div");
        setcolor(RED);
        rectangle(380,300,420,330);
        setfillstyle(SOLID_FILL,RED);
        floodfill(400,320,RED);
        setcolor(LIGHTGRAY);
        rectangle(384,304,416,326);
        setfillstyle(SOLID_FILL,LIGHTGRAY);
        floodfill(390,310,LIGHTGRAY);
        setcolor(RED);
        outtextxy(386,310,"Exit");
}
void main()
{
        display();
        detectmouse();
        showmousegraphics();
        detect();
}
```

16. Bitwise Operations

Computer memory consists of a series of bits, which may on or off, 1 or 0 respectively. A number is represented in a computer by a series of bits - giving the number in binary (base 2). Each digit tells us whether the relevant power of 2 is present.

Example - 1:

1. $165 = 1 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5 + 0 \cdot 2^6 + 1 \cdot 2^7 = 10101001$ (bin)

2. $108 = 0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5 + 1 \cdot 2^6 + 0 \cdot 2^7 = 01101100$ (bin).

3. $217 = 1 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6 + 1 \cdot 2^7 = 11011001$ (bin).

In a computer bits are usually grouped together. Here are some standard sizes:

Name	# of bits	# of Bytes	Range in Deci	Range in Hex
Nibble	4	½	0-15	0-0xF
Byte	8	1	0-255	0-0xFF
Word	16	2	0-65,535	0-oxFFFF
Double Word	32	4	0-4,294,967,295	0-OxFFFFFFFF

1.2 Hexadecimal

It is inconvenient to attempt to work in binary, so we often use the compromise of Hexadecimal.

Hexadecimal is the term used to describe base 16 numbers. We use the digits 0-9 as usual, and the letters A - F for the numbers 10 - 15.

HEX	A	B	C	D	E	F
DECI	10	11	12	13	14	15

In C and we identify a hexadecimal number by prefixing it with 0x. Other languages sometimes append h or H to indicate a hexadecimal number.

Example - 2:

1. $0xA5 = 10 \times 16 + 5 = 165 = 10100101$
2. $0x6C = 6 \times 16 + 12 = 108 = 01101100$
3. $0xD9 = 13 \times 16 + 9 = 217 = 11011001$

Note that each hex digit is a nibble (4 bits).

0xD = 1101 (bin) and 0xD0 = 11010000 (bin).
Thus there is a nice correspondence between hexadecimal and binary.

You should learn to think in Hex!

2 Bitwise Operations

When working with bytes we may preform the bitwise operations corresponding to each of the basic logical operations AND, OR & XOR.

To do this we preform the logical operation on each pair of bits from the two inputs from the same position. Taking 0 as False and 1 as True.

AND : Z = X & Y

X	0	1	0	1
Y	0	0	1	1
Z	0	0	0	1

Example And

108 & 217 = 0x6C & 0xD9 = 01101100 & 11011001 = 01001000
01001000 = 72 = 0x48
Note 0x6 & 0xD = 0x4 and 0xC & 0x9 = 0x8

OR: Z = X | Y

X	0	1	0	1
Y	0	0	1	1
Z	0	1	1	1

Example OR:

108 | 217 = 0x6C | 0xD9 =

01101100 | 11011001 = 11111101
11111101 = 253 = 0xFD

Note 0x6 | 0xD = 0xF and 0xC | 0x9 = 0xD

XOR: Z= X ^ Y

X	0	1	0	1
Y	0	0	1	1
Z	0	1	1	0

Example : XOR

108 ^ 217 = 0x6C ^ 0xD9 =
01101100 ^ 11011001=10110101
01101100 = 181 = 0xB5

Note 0x6 ^ 0xD = 0xB and 0xC ^ 0x9 = 0x5

NOT : Z = ~ X

X	0	1
Z	1	0

1's Complement

The bitwise operation corresponding to logical NOT is called 1's complement. Unlike &, |and ^, 1's complement is a unary operator (takes only one input).

~108 = ~ 0x6C = ~ 01101100=10010011
10010011 = 147 = 0x93

Note ~0x6 = ~0x9 and ~0xC = ~0x3

Most Computer Languages take 0 to be false, and any non zero value to be true. Generally if a C compiler is required to return a true value it will default to 1. Java has a Boolean data type.

Note that there is a difference between bitwise and, or, xor and 1's complement and the corresponding logical operations.

In order to highlight this difference languages like C and Java use different symbols for bitwise versus logical operations.

Operation	BitWise	Logical
AND	&	&&
OR	|	||
1's Comp./NOT	~	!

Note that according to the official C and Java specification, if the first part of a logical AND (&&) operation is false the second part is not guaranteed to be executed. Preforming '&' on a Java Boolean type guarantees that both will be executed.

Example - 3:

1. 0x6C & 0xD9 = 0x48, BUT
0x6C && 0xD9 = 1
(T ^ T = T).

2. 0x6C | 0xD9 = 0xFD, BUT
0x6C | | 0xD9 = 1
(T _ T = T).

3. _ 0x6C = 0x93, BUT
!0x6C = 0
(NOT T = F).

Finally note that in C and Java there is a difference between '=' and '=='. The first '=' is the assignment operator, whereas the second '==' is a logical operator. a = 2; means set the value of the variable a to 2. a == 2; means test whether the variable a is 2, return true if it is, and false otherwise.

Masking

Bitwise operations are particularly useful for masking. In this case each bit in a byte represents a value which may be either on or off, i.e. true or false. In this case we wish to be able to access the bits individually, to turn each bit on or off as desired.

This is particularly common when accessing hardware devices at a low level.

We can turn a bit on by a bitwise OR (|) with 1 in the relevant position.

We can test whether a bit is set by a bitwise AND (&) with 1 in the relevant position.

We can turn a bit off by a bitwise AND (&) with NOT (~) 1 in the relevant position.

We can toggle (on or off) a bit by a bitwise XOR (^) with 1 in the relevant position.

Example - 4:

Suppose we have a variable byte, in each case initially byte = 11001001, and we wish to manipulate the third bit:

1. byte | 00000100 (bin) turns the third bit on, byte = 11001101 (bin).

2. test = byte & 00000100 (bin),
test will be non zero (true) if the third bit was on in byte, otherwise it will be 0 (false).

3. byte & (~ 00000100 (bin)) turns the third bit off, ~ 00000100 = 11111011 (bin),
byte & 11111011 = 11001001 (bin).

4. byte ^ 00000100 (bin) toggles the third bit
byte = 11001101 (bin).
Here are the mask values in hex corresponding to the bits in a byte.

Bit	Mask Value (Hex)
7	0x80
6	0x40
5	0x20
4	0x10
3	0x8
2	0x4
1	0x2
0	0x1

Note that it is standard to count the first bit as bit 0. If we want to mask two bits we add the corresponding values. So to mask bits 1, 2, 5 and 7, we mask with 0xA6.

Example - 5:

The printer on many PCs is connected through the Parallel Port. This port has a (configurable) address of 0x378. This means that data is written and read at this address. The next byte, 0x379, contains the parallel port status register. Reading this byte gives the current printer status. The bits of the status byte have the following meanings Bit Printer Status

7	Busy
6	Acknowledge
5	Out of paper
4	Selected
3	I/O error

So if an application wishes to test if the printer is out of paper it will read the status byte into the variable StatusByte and test if bit 4 is set:
inb(StatusByte, 0x379); // Read Parport Status byte if(StatusByte & 0x10) ... // Check if bit 4 is set On the other hand a check for a busy printer: read(StatusByte, 0x379); // Read Parport Status byte if(StatusByte & 0x80) ... // Check if bit 7 is set

Example - 6:

The BIOS (Basic Input Output Services) controls low level I/O on a computer. When a computer first starts up the system BIOS creates a data area starting at memory address 0x400 for its own use. Address 0x417 is the Keyboard shift flags register, the bits of this byte have the following meanings:

Bit	**Value**	**Meaning**
7	0/1	Insert off/on
6	0/1	CapsLock off/on
5	0/1	NumLock off/on
4	0/1	ScrollLock off/on
3	0/1	Alt key up/down
2	0/1	Control key up/down
1	0/1	Left shift key up/down
0	0/1	Right shift key up/down

This byte can be written as well as read. Thus we may change the status of the CapsLock, NumLock and ScrollLock LEDs on the keyboard setting the relevant bit.

Each program has its own data and code segments which is allocated by the compiler. The normal pointer or near pointer can address locations which are in its data segment. **far pointer is used to address locations which are outside the data segment.**

Program to set on off the caps lock, scrolllock, numlock.

```
#include"stdio.h"
#include"conio.h"

void main()
{
        char ch;
        char far *memory1=(char far *)0x417;
        char far *memory2=(char far *)0x418;
        clrscr();
        printf(" \nPress  'C'  CapsLock on/off");
        printf("\n Press ‘N’ NumLock on/off”);
        printf("\n Press ‘S’ ScrollLock on/off”);
        printf(“\n Press  <ESC> - Quit");
        while(1)
        {
                ch=getch();
                if(ch==0x1b)
                break;
                switch(ch)
                {
                        case 'c':
                        case 'C':
                        *memory1=*memory1 ^ 64;
                        break;
                        case 'n':
                        case 'N':
                        *memory1=*memory1 ^ 32;
                        break;
                        case 's':
                        case 'S':
                        *memory1=*memory1 ^ 16;
                        break;
                }
        }
}
```

Exercises

1. Translate the following hexadecimal numbers to binary an decimal.
0x0, 0x10, 0xF, 0x1F, 0xA4, 0xFF

2. Find the bitwise and, or and xor of the following:
(a) 0xC6 with 0x35
(b) 0x19 with 0x24
(c) 0xD3 with 0xC7
(d) 0x17 with 0xFF

3. Find the 1's complement of the following:
0xC6, 0x35, 0xD3 and 0xC7.

4. In this question & is bitwise and, | is bitwise or, ^ is bitwise xor, and ! is 1's complement. a is any given two digit hexadecimal number. Explain why each of the following identities holds.

(a) 0xFF & a = a. (0xFF is the identity for AND)
(b) 0xFF | a = 0xFF. (0xFF is the absorbent for OR)
(c) 0xFF ^ a = !a
(d) 0 & a = 0. (0 is the absorbent for AND)
(e) 0 | a = a. (0 is the identity for OR)
(f) 0 ^ a = a. (0 is the identity for XOR)
(g) a ^ a = 0 (a is its own inverse under XOR)
(h) For any three two digit hexadecimal numbers a, b and c:
If a ^ b = c then a ^ c = b.

17. C++ (C WITH CLASS)

c-program

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
struct student
{
        long int rollno;
        char na[20];
        int t,e,m;
}stu;
void getdata()
{
        cout << "Enter Rollno";
        cin >> stu.rollno;
        cout << "Enter Name";
        cin >> stu.na;
        cout << "Enter 3 subject marks";
        cin >> stu.t >> stu.e >> stu.m;
}
void displayresult()
{
        if (stu.t>=35 && stu.e>=35 && stu.m>=35)
        cout << "Pass";
        else
        cout << "fail";
}
void main()
{
        getdata();
        displayresult();
}
```

c++ Program

```
#include <stdio.h>
#include <iostream.h>
class student
{
        private:
        long int rollno;
        char na[20];
        int t,e,m;
        public:
        void getdata()
        {
                cout << "Enter Rollno";
                cin >> rollno;
                cout << "Enter Name";
                cin >> na;
                cout << "Enter 3 subject marks";
                cin >> t >> e >> m;
        }
        void displayresult()
        {
                if (t>=35 && e>=35 && m>=35)
                cout << "Pass";
                else
                cout << "fail";
        }
};
void main()
{
        student ob;
        ob.getdata();
        ob.displayresult();
}
```

18. CLASS AND OBJECT

1. Class is a user defined data type
2. Class contains collection of variables and functions.
3. Class variables are called member variables and class functions are called member functions.
4. Member variables and functions have three scopes private, protected and public.
5. Inside a class scopes are meaningless, that is any member function can access any other functions or variables.
6. A variable created by a class is called object.
7. Object has the right to access only public members.
8. The default scope of a class member is private

Syntax

```
class classname
{
  scope:
        member declerations;
  scope:
        member declerations;
        ..
};
```

Example - 1:

```
#include <iostream.h>
#include <conio.h>
 class box
{
        private:
        int h,w,d;
        public:
        void getdata()
        {
                cout << "Enter value for height,width and depth";
                cin >> h >> w >> d;
        }
        void volume()
        {
                cout << "Volume = " << (h * w * d);
        }
 };
 void main()
 {
          box ob;
          ob.getdata();
          ob.volume();
 }
```

note : ob.h, ob.w, ob.d - are wrong object cannot access private members
ob.getdata(), ob.volume() - are right

Example - 2:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
void getdata()
{
        char na[20];
        cout << "Enter your name";
        cin >> na;
        if (strcmp(na,"admin")==0)
        {
                cout << "Enter value for height,width and depth";
                cin >> h >> w >> d;
        }
        else
        {
                cout << "Wrong user name";
        }
}
```

Replacing this function in the previous program, it allows only admin to entry the data, others should not allowed.

Member Function with Arguments

Example - 3:

```
#include <iostream.h>
#include <conio.h>
class box
{
        private:
        int h,w,d;
        public:
        void getdata(int a)
        {
                h = d = w = a;
        }
        void volume()
        {
                cout << "Volume = " << (h * w * d);
        }
};
void main()
{
         box ob;
         ob.getdata(20);
         ob.volume();
}
```

Member Function Overloading

Example - 4:

```
#include <iostream.h>
#include <conio.h>
class box
{
        private:
        int h,w,d;
        public:
        void getdata()
        {
                h = w= d = 0;
        }
        void getdata(int a)
        {
                h = d = w = a;
        }
        void getdata(int a, int b, int c)
        {
                h = a;
                w = b;
                d = c;
        }
        void volume()
        {
                cout << "Volume = " << (h * w * d);
        }
  };
void main()
{
         box ob1,ob2,ob3;
         ob1.getdata();
         ob2.getdata(20);
         ob3.getdata(10,20,30);
         ob1.volume();
         ob2.volume();
         ob3.volume();
}
```

Member Functions Declare Outside A Class Using Scope Resolution Operator (::)

Example-5

```
#include <iostream.h>
#include <conio.h>
class box
{
        private:
        int h,w,d;
        public:
        void getdata();
        void getdata(int a);
        void getdata(int a, int b, int c);
        void volume();
};
void box ::getdata(int a)
{
        h = d = w = a;
}
void box::getdata(int a, int b, int c)
{
        h = a;
        w = b;
        d = c;
 }
 void box::getdata()
{
        h = w= d = 0;
}
void box::volume()
{
       cout << "Volume = " << (h * w * d);
}
void main()
{
         box ob1,ob2,ob3;
         ob1.getdata();
         ob2.getdata(20);
         ob3.getdata(10,20,30);
         ob1.volume();
         ob2.volume();
         ob3.volume();
}
```

Inline Function

If a member function declare outside a class then prefix that function with inline keyword. When the class compile, Inline functions will be replaced in the actual place,

```
 inline void box ::getdata(int a)
{
        h = d = w = a;
}
```

Object As Member Function Arguments / Return type

Example - 6:

```
#include <iostream.h>
#include <conio.h>
 class box
{
        private:
        int h,w,d;
         public:
        void getdata()
        {
                cout << "Enter value for height,width and depth";
                cin >> h >> w >> d;
        }
        box sum(box ob1)
        {
                box x;
                x.h = this.h + ob1.h;
                x.w = this.w + ob1.w;
                x.d = this.d + ob1.d;
                return (x);
        }
        void volume()
        {
                cout << "Volume = " << (h * w * d);
        }
};
void main()
{
         box ob1,ob2,ob3;
         ob1.getdata();
         ob2.getdata();
         ob3 = ob1.sum(ob2);
         ob1.volume();
         ob2.volume();
         ob3.volume();
}
```

Friend Function

An outside function not a member function of a class can access the private members, if it is friend to a class.

Example - 7:

```
#include <iostream.h>
#include <conio.h>
class box
{
        Private;
        int h,w,d;
        public:
        void getdata()
        {
                cout << "Enter value for height,width and depth";
                cin >> h >> w >> d;
        }
        void volume()
        {
                cout << "Volume = " << (h * w * d);
        }
        friend box sum(box ob1, box ob2);
};
box sum(box ob1,box ob2)
{
        box x;
        x.h = ob1.h + ob2.h;
        x.w = ob1.w+ ob2.w;
        x.d = ob1.d + ob2.d;
        return (x);
}
void main()
{
         box ob1,ob2,ob3;
         ob1.getdata();
         ob2.getdata();
         ob3 = sum(ob1,ob2);
         ob1.volume();
         ob2.volume();
         ob3.volume();
}
```

Static Members

- Static members are global declaration to a Class
- Static functions or variables can access with class name or object name.
- Static functions can access only static variables.
- Static variables can also call from member functions.

Example - 8:

```
#include <iostream.h>
#include <conio.h>
class static1
 {
        public:
        static int i;
        int j;
        static void displayi()
        {
                cout << " i = " << i;
        }
        void getdata()
        {
                cout << "Enter a value";
                cin >> j;
                i++;
        }
        void display()
        {
                cout << " i = " << i;
                cout << " j = " << j;
        }
};
int static1::i;
void main()
{
        static1::i = 10;
        static1::displayi();
        static1 ob;
        ob.getdata();
        ob.display();
}
```

19. CONSTRUCTOR FUNCTION

- Constructor functions execute automatically when create object.
- Constructor function name and class name should be same
- Constructor function does not have return type.
- Purpose to initialize values to member variables.

Example : Refer Example - 9

Example - 9:

```
#include <iostream.h>
#include <conio.h>
class box
{
        private:
        int h,w,d;
        public:
        box()
        {
                h = w = d = 0;
        }
        box(int  i)
        {
                h = w=d=i;
        }
        box(int i, int j, int k)
        {
                h = i;
                w = j;
                d = k;
        }
        void volume()
        {
                cout << "volume = " << (h * w * d);
        }
};
void main()
{
        box ob1,ob2(10),ob3(5,10,15);
        ob1.volume();
        ob2.volume();
        ob3.volume();
}
```

Example - 10:

```
#include <iostream.h>
#include <conio.h>
 void main()
 {
        box ob1,ob2,ob3;
        ob1 = box(10);
        ob2 = box(5,10,15);
        int k;
        cout << "Enter a value";
        cin >> k;
        ob3 = k;  // equal to ob3 = box(k);
        ob1.volume();
        ob2.volume();
        ob3.volume();
}
```

Empty Constructor

If we do not provide any constructor for a class then the empty constructor will be added by c compiler

format classname(){}

note: If user add a constructor, then compiler will not add the empty constructor.

Constructor Call Explicitly / Copy Constructor

Constructor can also call explicitly after the object created.

Example : Refer Example - 10

20. DESCTRUCTOR FUNCTION

- Executes automatically when an object destroyed.
- Same as constructor rules but the destructor name prefix with ~ sign.

Example - 11:

```
#include <iostream.h>
#include<conio.h>
class box
{
        private:
        int h,w,d;
        public:
        box()
        {
                h = w = d = 10;
        }
         void volume()
        {
                cout << "volume = " << (h*w*d);
        }
         ~box()
        {
                cout << "object destroyed";
        }
};
 void main()
 {
        box ob;
        ob.volume();
 }
```

output

```
volume = 1000
object destroyed
```

21. OPERATOR OVERLOADING

Operators do not work with object. Make it to work operator overloading functions are used.

box ob1,ob2,ob3;
ob3 = ob1 + ob2; - wrong
(Make it to work need operator overloading function)

1.Unary Overloading

Operators come with one object.

```
void operator op()
{
        statements
}
```

op – any operator (+, -, >>, << etc…)

Example : Refer Example – 12

2.Binary Overloading Using Member Function

Operators come between 2 elements. In that the left side of the operator is object and the right side of the operator is object or any other data type member.

```
return type operator op(one argument)
{
  statements;
}
```

Example : Refer Example – 13

Example - 12:

```
#include <iostream.h>
#include<conio.h>
class box
{
        private:
        int h,w,d;
        public:
        box()
        {
                h = w = d = 10;
        }
        void operator -()
        {
                h = -h;
                w = -w;
                d = -d;
        }
        void volume()
        {
                cout << "\nVolume = " << (h*w*d);
        }
};
void main()
{
         box ob;
        ob.volume();
        -ob;
        ob.volume();
}
```

Output:

Volume = 1000
Volume = -1000

Example - 13:

```
#include <iostream.h>
#include <conio.h>
class box
{
        private:
        int h,w,d;
        public:
        box()
        {
                h = w = d = 10;
        }
        box operator + (box ob)
        {
                box ob1;
                ob1.h = h + ob.h;
                ob1.w = w + ob.w;
                ob1.d = d + ob.d;
                return (ob1);
        }
        box operator + (int k)
        {
                box t;
                t.h = h + k;
                t.w = w + k;
                t.d = d + k;
                return (t);
        }
        void volume()
        {
                cout << "volume = " << (h*w*d);
        }
};
void main()
{
        box ob1,ob2,ob3;
        ob3 = ob1 + ob2;
        ob1.volume();
        ob2.volume();
        ob3.volume();
        ob3 = ob1 + 20;
        ob3.volume();
 }
```

note:

```
        ob3 = ob1 + ob2;
        ob3 = ob1.operator + (ob2);
        ob3 = ob1 + 20;
        ob3 = ob1.operator +(20);
```

3. Binary Overloading Using Friend Function

Operators come between 2 elements. In that the left and right side of the operator is send as arguments.

Example – 14:

```
#include <iostream.h>
#include <conio.h>
class box
{
        private:
        int h,w,d;
        public:
        box()
        {
                h = w = d = 10;
        }
        void volume()
        {
                cout << "volume = " << (h*d*w);
        }
        friend box operator + (box ob1,box ob2);
        friend box operator + (int x, box ob2);
};
box operator + (box ob1, box ob2)
{
        box ob;
        ob.h = ob1.h + ob2.h;
        ob.d = ob1.d + ob2.d;
        ob.w = ob1.w + ob2.w;
        return (ob);
}
box operator + (int x, box ob2)
{
        box ob;
        ob.h = x + ob2.h;
        ob.d = x + ob2.d;
        ob.w = x + ob2.w;
        return (ob);
}
void main()
{
        box ob1,ob2,ob3;
        ob3 = ob1 + ob2;
        ob3  = 10 + ob2;
        ob1.volume();
        ob2.volume();
        ob3.volume();
}
```

Note : friend binary function call as

```
ob3 = ob1 + ob2;
    = operator + (ob1, ob2);
```

Example:

```
ob3 = 10 + ob1;
ob3 = ob1 + 20;
```

To execute these statements following friend operator overloading functions are required.

```
box operator + (int a, box ob) ;
box operator + (box ob, int a);
```

Example programs for Overloading >> and << operators:

Example Program – cout << ob;
Example Program – cin >> ob;

Example - 15:

```
#include <iostream.h>
#include <conio.h>
class box
{
        public:
        int h,w,d;
        box()
        {
                h = w = d = 0;
        }
        friend void operator << (ostream &cout1, box ob1);
        friend void operator >> (istream &cin1, box &ob1);
};
void operator << (ostream &cout1, box ob1)
{
        cout1 << ob1.h << ob1.w << ob1.d;
 }
void operator >> (istream &cin1, box &ob1)
{
        cin1 >> ob1.h >> ob1.w >> ob1.d;
}
 void main()
{
        box ob1;
        clrscr();
        cin >> ob1;
        clrscr();
        cout << ob1;
}
```

3. Operator Type Function

```
operator datatype()
{
        statements;
}
```

Executes automatically when the left side of an assignment statement is predefined data type and right side is object.

Example - 16:

```
#include <iostream.h>
#include<conio.h>
class box
{
        public:
        int h,w,d;
        box()
        {
                h = w = d = 10;
        }
        operator int()
        {
                return (h*w*d);
        }
};
void main()
{
        box ob;
        int vol;
        vol = ob;
        cout << "volume = " << vol;
}
```

22. INHERITENCE
(Merging 2 or more classes)

Syntax

```
class drive : public or private class base,
              public or private class base1
{
        drive members;
}
```

1. Class members declare with 3 scopes private, protected or public.
2. Private members cannot inherit
3. Public or protected members can inherit.
4. Inheritance has 2 scopes public and private
5. If the inheritance scope is public then all the base protected and public members added with drive class members as it is in the base class.
6. If the inheritance scope is private then all the base class members first convert as private and then added to drive class, hence further inheritance is not possible.

Inheritance Has Four Types

Example:

class a class b class c class d class e

1. Single Inheritance

class b : public a

here : b = b + a

2. Multilevel Inheritance

```
class b : public  a
class c : public b
```

Here: b = b + a
       C = c + b

3. Multiple Inheritance

```
class c : public a, public b
```

Here again: c = c + a + b

4. Hybrid Inheritance

```
class b : public a
        b = b + a
class c : public d
        c = c + d
class e : public b, public c
```

Here: e = e + b (b +a) + c (c + d);

Example : Refer Example - 17

class b : public a

Now Drive Class B has the following members

```
private:
        int bpri;
protected:
        int pro1,pro2,bpro1;
public:
        int pub1, bpub1;
```

class b : private a

then b has the following members

```
private:
        int bpri,pro1,pro2,pub1;
protected:
        int bpro1;
public:
        int bpub1;
```

Virtual Base Class

In hybrid inheritance to avoid a class inherit more than once, virtual base class is used.

```
class a
class b
class c
class d

class b : virtual public a
class c : virtual public a
class d : public b, public c
```

here class d = d + b + c + a

Inherit class a only once.

Constructor Call in Inheritance

In inheritance derived class constructor should execute the base class constructor.

Example : Refer Example - 18

Example - 17:

```
class a
{
        private:
        int pri;
        protected:
        int pro1, pro2;
        public:
        int pub1;
};
class b
{
        private:
        int bpri;
        protected:
        int bpro1;
        public:
        int bpub1;
}
```

Example - 18:

```
#include <iostream.h>
#include <conio.>
class a
{
        int i;
        public:
        a(int a1)
        {
                i = a1;
                cout << "class a executes";
        }
 };
class b : public a
{
        int j;
        public:
        b(int a1, int b1):a(a1)
        {
                j = b1;
                cout << "class b executes";
        }
  };
class c : public b
{
        int k;
        public:
        c(int a1, int b1, int c1):b(a1,b1)
        {
                k = c1;
                cout << "class c executes";
        }
};
void main()
{
            c ob(10,20,30);
}
```

23. POINTERS AND VIRTUAL FUNCTION

C- Program

int a, *b;

To Assign Address to a pointer variable

```
b = &a;
b = (int *)malloc(2);
```

To release pointer memory location

```
Free (b);
```

C++ - Program

int *a,*b;

To allocate memory use new keyword

```
a = new int;
b= new int[5];
```

To release pointer memory location

```
delete a;
delete [] b;
```

Example - 19:

```
#include <iostream.h>
#include <conio.h>
class box
{
        private:
        int h,w,d;
        public:
        void getdata()
        {
                cout << "Enter value for h,w,d";
                cin >> h >> w >> d;
        }
        void volume()
        {
                cout << "Volume = " << (h*w*d);
         }
};
void main()
{
        box ob1,*pt;
        ob1.getdata();
        ob1.volume();
        pt = &ob1;
        pt->volume();
        pt = new box();
        pt->getdata();
        pt->volume();
}
```

Example - 20:

```
#include <iostream.h>
#include <conio.h>
class box
{
        public:
        int a;
        void geta()
        {
                cout << "Enter value for a";
                cin >> a;
        }
        void displaya()
        {
                cout << "value of a = " << a;
        }
};
class box1 : public box
{
        public:
        int b;
        void getb()
        {
                cout << "Enter value for b";
                cin >> b;
        }
        void displayb()
        {
                cout << "value of b = " << b;
        }
};
```

class box has three members

a
geta()
displaya()

class box1 has six members

a
b
geta()
displaya()
getb()
displayb()

box ob1, *pt1;
box1 ob2, *pt2;

pt1 = &ob1;
pt2 = &ob2;

ob1.a	pt1->a
ob1.geta()	pt1->geta
ob1.displaya()	pt1->displaya

ob2.a	pt2->a
ob2.b	pt2->b
ob2.geta()	pt2->geta
ob2.displaya()	pt2->displaya
ob2.getb()	pt2->getb
ob2.displayb()	pt2->displayb

pt1 = &ob2;

Base pointer can point to drive class but base pointer can access only base members available in that drive class.

That is pt1 can access

pt1->a
pt1->geta
pt1->displaya

and pt1 cannot access

pt1->b
pt1->getb
pt1->displayb

Example - 21:

```
#include <iostream.h>
#include <conio.h>
class base
{
 public:
        void show()
        {
                cout << "base class";
        }
};
class drive : public base
{
        public:
        void show()
        {
                cout << "drive object";
        }
};
void main()
{
        base *pt;
        drive ob1;
        pt = &ob1;
        pt->show();
}
```

Output:

base class

Example – 22:

```
#include <iostream.h>
#include<conio.h>
class base
{
        public:
        virtual void show()
        {
                cout << "base class";
        }
};
class drive : public base
{
        public:
        void show()
        {
                cout << "drive object";
        }
};
void main()
{
        base *pt;
        drive ob1;
        pt = &ob1;
        pt->show();
}
```

Output:

drive object

Virtual Function

A function with same name, arguments and return type in base as well as in drive class, then the base pointer executes the drive function in place of base function if the base class function is virtual type.

Example : Refer Example - 22

24. ABSTRACT CLASS

1. Abstract class contains prototype functions.
2. Prototype functions have the function heading without function code.
3. Abstract class is used in inheritance as well as pointer creation.
4. Object cannot create by using abstract class.
5. The class which inherits abstract class should provide code for the prototype function.

Example - 23:

```
#include <iostream.h>
#include <stdio.h>
class mouse
{
        virtual void mouseclick()=0;
        virtual void mouseup()=0;
};
class drive : public mouse
{
        void mouseclick()
        {
                cout << "Mouse Click";
        }
        void mouseup()
        {
                cout << "Mouse Up";
        }
};
void main()
{
          mouse *pt;
        drive ob;
        pt = &ob;
        pt->mouseclick();
        pt->mouseup();
}
```

Output:
Mouse Click
Mouse Up

25.MORE FUNCTIONS WITH CIN AND COUT

Use #include <iomanip.h>

1, cin.get()
Input a char

ch = cin.get() equal to getch()

2,cout.put(ch)
Display a char

3.cin.read(stringvaraible, numberofchars)
4.cin.getline(stringvar, numberofchars)
5.cout.write(stringvariable,numberofchars)

Input and output number of character from the user.

Example - 24:

```
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
#include <string.h>
void main()
{
        char na[20];
        cout << "Enter a Name";
        cin.getline(na,20);
        for(int i = 1;i<= strlen(na);i++)
        {
                cout.write(na,i);
                cout << endl;
        }
        cin.get();
}
```

Output:
Enter a Name : Welcome
w
we
wel
welc
welco
welcom
welcome

Formatted Output

1.cout.width(n)
2. cout << setw(n);

These two functions display data in a specific width.

Example - 25:

```
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
void main()
{
        clrscr();
        cout << "\n";
        cout.width(10);
        cout << "SLno";
        cout.width(20);
        cout << "Company Name";
        cout.width(20);
        cout << "Address";
        cout << "\n";
        cout.width(10);
        cout << "1";
        cout.width(20);
        cout << "IWS Coimbatore";
        cout.width(20);
        cout << "Ram Nagar";
        cout << endl;
        cout << setw(10) << "2" << setw(20) << "IWS Noida" << setw(20) << "New Delhi";
        cin.get();
}
```

output

slno	name	address
1	IWS Coimbatore	Ram Nagar
2	IWS Noida	New Delhi

26. TEMPLATE FUNCTION

Use to avoid function overloading concept.

Example:

```
void big(int a, int b)
{
        If (a > b)
        cout << "big=" << a;
        else
        cout << "big=" << b;
}
void big(float a, float b)
{
        if (a > b)
        cout << "big =" << a;
        else
        cout << "big = " << b;
}
```

Write one function for the above 2 functions using function template.

Example - 26:

MP

Note: First function call replaces the T with int and the second function replaces the T with float.

Appendix - A

Algorithm and Flowcharts

What Is an Algorithm?

A step by step procedure to solve a particular problem. It consists of English like statements. Each statement must be precise and well-defined to perform a specific operation.

Characteristics of an Algorithm

Each and every algorithm is characterized by the following five important characteristics.

1. Input – It may accept a zero or more inputs
2. Output – It should produce at least one output (result)
3. Definiteness – Each instruction must be clear, well-defined and precise. There should not be any ambiguity.
4. Finiteness: It should be a sequence of finite instructions. That is, it should end after a fixed time. It should not enter into an infinite loop.
5. Effectiveness: This means that operations must be simple and are carried out in a finite time at one or more levels of complexity. It should be effective whenever traced manually for the results.

Algorithmic Notations:

1. **Name of the algorithm:** It specifies the problem to be solved.
2. **Step number:** Identification tag of an instruction and it is an unsigned positive integer.
3. **Explanatory comment:** It follows the step number and describes the operation. It should be written within a pair of square brackets.
4. **Termination:** It specifies the end of the algorithm. It is generally a STOP statement and the last instruction in the algorithm.

Example 1: Write an algorithm to compute the area of the circle.

Algorithm: Area of a circle

Step 1:	**Read radius**
Step 2:	[Compute the area]
	Area = 3.142 x radius x radius
Step 3:	[print the area]
	Print "Area of a circle = ", Area
Step 4:	[End of algorithm]
	Stop

Example 2: Write an algorithm to find the largest of three numbers

Algorithm: Largest of three numbers

Step 1:	[Read the values of A, B and C]
	Read A,B,C
Step 2:	[Compare A and B]

If (A > B) goto step 4

Step 3: [Otherwise compare B with C]
If [B > C] then
Print "B is the Largest"
Else
Print "C is the largest"
Goto Step 5

Step 4: [Compare A and C for largest]
If [A > C] then
Print "A is the Largest"
Else
Print "C is the largest"

Step 5: [End of the algorithm]
Stop

Example 3: Write an algorithm to compute the sum of first N natural numbers and their means value. The natural numbers are 1,2,3,…N. It adds one number at a time until N, the maximum limit. Then computes the Means of them.

Algorithm: Sum and Mean of Natural Numbers

Step 1: [Read the maximum value of N]
Read N

Step 2: [Set sum equal to 0]
Sum = 0

Step 3: [Compute the sum of all N numbers]
For Num = 1 to N step 1
Sum = sum + num
End for

Step 4: [Compare mean value of N natural numbers]
Mean = sum / N

Step 5: [Print Sum and Mean]
Print "sum of N natural numbers = ",sum
Print "Mean of N natural number = ", mean

Step 6: [End of program]
Stop

Pseudo code

A pseudo code is neither an algorithm nor a program. It is an abstract form of a program It consists of English like statements.

Example

```
Read n1,n2
Sum = n1 + n2
Diff = n1 – n2
Mult = n1 * n2
Quot = n1 / n2
Print sum, diff, mult, quot
End
```

Coding

The complete structure of a problem to be solved by a computer is called a program. The computer does not process an algorithm or a flowchart, but executes the program.

Debugging

The process of detecting and correcting errors in the program is known as debugging.

Debugger

Each language has its own debugger. Debugger is a program that takes an object program as input and executes it and helps in eliminating the mistakes that occur in the source program.

Programming Errors:

Generally, programmers do three types of errors. They are Syntax errors, Logical errors and run-time errors

Syntax Errors

These types of errors are violation of the grammar or rules of programming languages.

Logical Errors

Logical errors occur during the coding process. The program will be executed but produce some unwanted results. It is very difficult to debug such errors, because the compiler or debugger does not display them. We can eliminate such errors by tracing it and running the sample data.

Run-Time Errors

Run time errors mostly device errors, errors in system software, keypunch errors, incorrect data input etc. Some of the Run time errors

Divide by zero, Null pointer assignment and
Data Overflow

Testing

The process of executing the program to test the correctness of the outputs (results) of the problem is called testing. The program is tested by executing with different sets of data. Logical errors are the outcome of this process.

Documentation

While writing programs, it is a good programming practice to make a brief explanatory note on the program or project segments. This explanatory note is called a comment. It explains how the program works and how to interact with it. Thus, it helps the other programmers to understand the program.
There are two types of documentation. They are : Internal documentations
External documentation

Internal documentation

This documentation is a comment statement within a program. Explain the variable purpose, describes the functions of the program or program segments. These statements are not translated to machine language.

External documentation

This documentation is an executable statement in a program. It may be a message to the user to respond to the program requirement. Mostly this is accomplished using output statements.

Examples:

Print “Sum of Add values’, add
Print “ 5 factorial value = “, fact

Appendix – B

Write an Algorithm and Flowchart for the following

1. To perform the basic arithmetic operations such as addition, subtraction, multiplication and division.
2. To calculate the simple interest and compound interest for the amount deposited (P) for some years (T) for the rate of interest (R)
3. SI = (P * R * T)/100
4. CI = P X (1 + R/100) ^T - P
5. To reverse a given integer number.
6. To compute the factorial of a given number
7. find the average of marks obtained in three tests.
8. Compute volume of a cube
9. Convert Fahrenheit temperature into Celsius
10. Find the sum of digits of a given number
11. Interchange two numbers without using the third variable
12. Generate 50 Fibonacci sequence numbers.
13. Find the sum of even and odd numbers between 1 and 100
14. Generate all prime numbers between two numbers
15. Reverse the given number and check whether it is a palindrome or not
16. Generate all numbers which are divided by 3 and not by 7
17. Convert decimal into binary
18. Convert binary into decimal
19. Find out sum of 1 - ½ + 1/3 - 1/4 …..
20. Find out 1 /2 – 2/3 + 3/4 – 4/5 ..
21. Find out $x + x^2 + x^3 + \ldots$
22. Find out $x - x^2 / 2! + x^3 / 3! + \ldots$
23. Simulate multiplication operation by addition operation only.

Appendix - C

Important Question and Answers

1. What is a computer?

A computer is an electronics device. It accepts data and instructions from the user, stores it in its memory, processes it and produces the desired output.

2. What are the main parts of a computer system?

A computer system has four important components. They are 1. Data 2. Software 3. Hardware and 4. User.

3. What is data?

Data is a basic thing that a computer process
It is a raw fact that the user can feed into the computer.
It does not have any implicit meaning.
Numbers, names, pictures and sound are examples of data
Examples 25, 567.48 etc are numbers; sreya, ganesh are names

4. What is information?

Information is a processed data
It has an implicit meaning
Example : Date : 01/08/2010, Name : Raja

5. What is Software?

Software is a program or a set of programs designed to help the users.
A software perform a specific task or a group of tasks
There are two types of software:
Application Software
System Software
We cannot touch and experience software
Ms-Word, Ms-Excel, Oracle etc are examples of software's

6. What is hardware?

Hardware is a physical components in a computer system
User can touch and see the hardware components
All peripheral devices are hardware units of a computer
Keyboard, mouse, monitor, pen drive, etc. are examples of hardware

7. What is a program?

A program is a set of instructions to solve a particular problem
A program may be simple or complex
A programming language is used to write a program

8. What are the four steps of information processing cycle?

Input, Output, Processing and Storage

9. What is a CPU?

CPU is acronym for Central Processing Unit of a computer
It is a microprocessor that accepts instructions, decodes and executes them

It has two important units called Control Unit and Arithmetic and Logical Unit (ALU)

10. What is a Control Unit and what are its functions?

Control Unit is an important unit in a computer. It controls and coordinates the activities of all the units of a computer system.
Control Unit performs the following functions
Fetching data and instructions from the main memory
Interpreting these instructions
Controlling the transfer of data to and from the main memory
Controlling input and output devices
The overall supervision of computer system.

11. What is an ALU?

ALU perform arithmetic operations like addition, subtraction, multiplication, division etc.
It also cafries out logical operations such as AND, OR and NOT etc.

12. **List the input and output devices which you can use and see in your computer lab?**

Input Devices : Keyboard, Mouse, Light pen, Joystick, Microphone, Disk
Output Devices: monitor, Plotter, Printer, Disk, Speaker, Data Projectors

13. What is a RAM?

RAM stands for Random Access Memory
It is the main memory of a computer system
It is a semiconductor memory
It is a volatile memory – as the contents stored on it are erased when the computer is switched off.

14. What is a ROM?

RAM stands for Read Only Memory
It is also main memory of a computer system
It is also a semiconductor memory
It is a permanent memory – as the contents stored on it are not erased when the computer it turned off.

15. What is a byte?

A byte is a sequence of 8 bits
It is the basic unit of computer memory
1024 bytes make up one kilobyte (KB), 1024 x 1024 will make one megabyte (MB) and similaryly 1024 x 1024 x 1024 bytes will make on Gigabyte (GB)

16. What is a word?

A word is a sequence of 16 – bits or a group of two bytes

17. What is a nibble?

A nibble is a sequence of 4 bits or a half –a –byte
Memory is not measured in nibbles

18. How computers represent data?

Computer's microprocessor is made up of transistors which work on two states ON and OFF. The ON state is represented by a 1 and the OFF state is represented by a 0. So, the data in computers is represented as s sequence 0s and 1s. Because of this reason, the binary number system is regarded as the language of computers.

19. Write the binary equivalent of decimal number 34?

The Binary equivalent of the decimal number 34 is:

0010 0010 (in 8 bit representations)

0000 0000 0010 0010 (in 16-bit representation)

20. Where do you store data and instructions?

Data and instructions are stored in FOUR different types of media. They are
Semiconductor media (RAM and ROM)
Magnetic media (Floppy, hard disk and magnetic tapes
Optical media (CD-ROM, DVD-ROM and Bluray disk)
Solid state media (Flash memory)

21. how do you store data and instructions?

A disk is made of a thin circular plastic coated with magnetic oxide. This circular plate has a set of concentric circle on its either surfaces. These circles are called tracks. And all the tracks are further divided into conical portions called sectors. The area between tracks and sectors stores the data.

22. How do you compute the disk capacity?

A floppy disk capacity is basically the storage capacity. It is computed by the following formula.

Capacity = No. of surfaces x No. of tracks x No. of sectors x bytes per sector.

23. What are the factors that affect the processors speed?

There are many factors which affect the speed of a processor. They are:
CPU registers, Memory (main) and Computing power, Bus, Cache memory.

24. What is clock speed?

The clock speed refers to the speed at which the computer executes instructions.

25. Name the three different types of buses used in the computers?

The three buses that are used in a computer system are:

Data Bus, Address Bus and Control Bus

26. Distinguish between a Level 1 cache and Level 2 caches?

The level 1 chache is the fastest memory of all. This memory is placed on the CPU chip. Therefore, it is called on chip cache. It is at the highest level of all memory hierarchy.

The Level 2 cache is the memory between the RAM and the CPU, BUT not directly on the CPU chip itself.

27. What do you mean by a boot sector?

A boot sector is a sector on a disk that contains a boot program which runs when the computer is switched on.

28. What is booting?

The process of starting up the operating system of a computer using its boot program is called booting.

29. What is an operating system?

An operating system is an integrated collection of programs which make the computer operation and help executing the user's programs. It acts as an interface between the user and the PC. It manages the computer system resources such as memory, processor, storage, input-output devices and files.

30. List the four functions of an operating system?

Memory Management: Allocating memory to the running programs and deallocation of memeory when the programs are terminated.

Processor Management: Processing the jobs(programs), deciding on the job scheduling technique and how log a job to be processed. And also releasing the processor when the jobs are terminated.

Device Management: aAllocating the input and output devices to the running processes and deallocating them when the processes are terminated.

File Management: Managing the file system. Opening and closing files, providing access permission to the files, keeping track of files with their status and memory locations.

31. What are the different types of operating systems?

Operating systems may be categorized into four types. They are

Single-user / Single-tasking operating system
Single-user / Multitasking operating system
Multi-user / Multitasking operating system
Real-time operating system

32. Who developed WWW and where was it developed?

WWW as developed in 1989 by Tim Berners-Lee of the European Particle Physics Lab (CERN) in Switzerland.

33. Give the salient feature of C Language?

C language is both high-level and low-level language
It is used for developing both application and system software
It has a rich set of operators and data types
Pointers are the powerful concepts that C supports.

34. What is an Identifier?

An Identifier is the name given to the elements of a C program. A variable, an array, a function, a pointer or a structure in C program are identifiers.

35. What is a Keyword?

A keyword is a reserved word in a programming language and whose meaning is predefined. Therefore a keyword cannot be used as an identifier. The int, float, void, enum, struct etc are some of the C keywords.

36. Mention the different data types of C language?

C supports basically three data types

Built-in data types
- Integral data types – int, char
- Floating-point data types – float, double
- Non-specific – Void

User- Defined data types
- Type definition – typedef
- Enumerated data types – enum
- Derived data types
- Arrays
- Pointers
- Structures and unions
- Functions etc.

37. What is a variable and how do you declare a variable in C?

A variable is an identifier in C. It designates some memory location at which the data is stored. A variable can be declared in C as follows.

```
Int a;
Float f1, fval;
Char ch;
```

38. Can you assign a value to a variable during compilation?

Yes, we can assign a value to a variable during compile-time. For example, if we want to assign a decimal number 10 to a variable ivar of type int, we can do it by declaring the following statement in the C program as

```
Int iwar = 10;
```

39. Explain the meaning of Lvalue?

Lvalue or location value is an address that is stored into. In the assignment statement

```
a = 10,
```

we have the lvalue of a used on the left hand side and the integer constant value 10 assigned to this location. In the assignment

```
b = a;
```

We have the rvalue of a (i.e. 10) used to assign to the lvalue of b. So depending on which side of an assignment, we use the variable a, we get two different behaviors.

40. What is the difference between an assignment and initialization?

Initialization is a process of assigning values to variable during compile-time. This is a onetime process. And it is done as part of variable declaration. For example.

```
Void main()
        {
                Float p = 32.15;
        }
```

Assignment, on the other hand, is a process of providing values to the variables during run-time (execution) This can be done any number of times. And it is done after the declarations. For example

```
        Void main()
        {
                Float p;
                P = 32.15;
        }
```

41. What are control structures?

Control structures are the statements help control the order of execution of statements in a program. They are useful to the programmers to incorporate the following:
Execution of a statement or a group of statements for a fixed number of times
Execution of a statement or a group of statements as long as a certain condition is true
Transfer the control from one point in a program to other
Termination of execution after a specified point
By pass certain statements during the execution of a program.

42. How do you categorize the control structures?

Conditional Control Structures
Branching statement
If, else if, nested-if, switch
Looping
For, while, do-while
Un conditional control structures
Un conditional jump – goto
Termination - break, return
Bypassing - continue

43. What is an Array?

An array is a data structure which hold elements of same data type.
It helps to process a group of homogeneous items under a single variable name.

44. How do you classify arrays?

One dimensional arrays - Array[]
Multi-dimensional arrays
Two-dimensional arrays – Array[row][col]
Three-dimensional arrays – Array[r1][r2][c1];

45. What is the output of the following program?

```
#include <stdio.h>
Void main()
{
        Int a[10];
        Int I;
        for (I = 0;I < 10;i++)
        {
            A[i] = I + 1;
        }
        Printf("%d   %d", a[4], 4[a]);
}
```

This program produces 5 5 as output. An array name designates the address of the first element in array. Therefore, by definition, a[4] is equivalent to *(a + 4) and 4[a] is equivalent *(4 + a).Remember commutative property of addition: A + B = B + A.

46. What are the points that one should remember while initializing a two-dimensional array?

The number of sets of initial values must be equal to the number of rows in the array (a 2-D array is a matrix).

One – to – one mapping is preserved. That is, the first set of initial values is assigned to the first row elements and the second set of initial values is assigned to the second row elements and so on.

If the number of initial values in each initializing set is less than the number of the corresponding row elements, then all such elements of that row are automatically assigned to zeros.
If the number of initial values in each initializing set exceeds the number of the corresponding row elements then there will be a compilation error.

47. What does the following program print?

```
#include <stdio.h>
Void main()
{
    Int  array[3] = {10,20,30};
    Int I;
    I = sizeof(array) / sizeof(array[0]);
    Printf(" I = %d",i);
}
```

This program prints the number of elements in a given array, so ans is : 3

48. What is a function?

A function is a set of instructions to solve a specific and well-defined problem.
A function is complete and independent. (It has its own set of instructions to execute, declaration, beginning and end).
A function may or may not return a value to the calling program after its execution. If the function to return the value to the calling program, then the return statement has to be used in the function definition in the function.

49. What are the benefits of using functions?

Reusability: A function, once written for solving a specific task, can be used again and again. We need not rewrite it rather reuse it.
Build own library: User can build his/her own library of the most commonly used functions. This reduces the time and space complexity. It also promotes the portability.
Debugging is easier: Since each function is a small module, it is easier to locate the errors and correct them.

50. Distinguish between function definition and function declaration?

The function definition and function declaration are different. The function definition means writing of the actual code. But the function declaration is the instruction given to the compiler regarding the functions that appear after the main() function in a C program.

51. What is a calling program and how is different from called program?

If a given program is divided into the number of small modules called functions, then there may be a communication between some of them.

A function which calls other function is called a calling function. And the one being called is called the called function. For example, if a function funcx() is called inside the body of function funcy(), then function funcy() is a calling program and funcx() is a called program.

52. If the specific return type is not mentioned, then what value will be returned?

If the return type is not mentioned, by default, the function returns an int value.

53. Distinguish between arguments and parameter?

Arguments are the variables which appear in the function call. These are the actual parameter whose values are passed to the functions. Parameters, on the other hand, are the variable, used In the function definition. They receive the values of the actual parameters for the specified computation. They are called dummy parameters.

54. What do you mean by parameter passing and list the different types of parameter passing mechanisms in C?

Parameter passing is a mechanism through which the values of the arguments are passed onto a function. The called function receives these arguments in the form of parameters (variables that hold the values of the arguments) and uses them of a specific computation.

There are two types of parameter passing mechanisms in C.

.

Call – by – Value
Call – by – reference

Call by Value	Call by Reference
In this method, the values of the arguments are passed from a calling function to a called function	In this method, the addresses of the arguments are passed from a calling function to a function
The values are copied into the called function	No values are copied into the called function
Any changes made to these values in the called function, will not impact the original values in the calling function	Any changes made to these values in the called function, will modify the original values in the calling function

55. Consider the following function. Determine what value would be returned for the call fun(10)?

```
Void fun(int x)
{
        If (x/2 == 0)
        Return 5.0;
        Else
        Return 0.0;
}
```

This function would return an error message, as the data type of the return value is void. But it is returning float value.

56. What is recursion?

Recursion is the name given to the technique of defining a function in terms of itself.
A recursive function calls and recalls itself, till it reaches a stopping condition.

Appendix – D

Assignment Questions

1. Explain the different steps involved in solving a problem by the computer
2. What is an algorithm? Explain its features
3. What is a flowchart?

Differentiate between

5. Algorithm and flowchart
6. Code and Pseudo code
7. Internal and external documentation
8. Syntax error and logical error

9. What is testing?
10. What are the silent features of C language?
11. What are the constituent parts of C program?
12. Explain the steps involved in compiling and executing a C program on DOS
13. What are C Tokens and Explain?
14. Distinguish between the keywords and identifiers
15. What is a constant? Explain different types of C Constants
16. What is a variable? When do you need a variable?
17. List the rules for forming variable names
18. Why do you declare variable
19. Differentiate between the constant and variable
20. What is data type? Explain four basic data types in 'C Language'
21. Distinguish between string constant and character constant
22. What is the range of int, float, and unsigned char for 16/32 bit processor?
23. What is the need of backslash constant?
24. What is a statement? What types of statements does C Supports
25. What is a symbolic constant? How are they defined?
26. Explain the concept of multiple assignment in C with example
27. Why do you need delimiters in the program? 28. Explain any 2 delimiters
29. Write the different between %s and %[^\n]
30. Write the difference between scanf() and gets() for string input
31. What is the use of modulus operator?
32. Define the terms
    - Priority (precedence)
    - Associativity
33. How do you classify C OPERATORS?
34. What are bitwise operators? Explain with Example
35. What is the need for type conversion? Explain with Example
36. What are header files? Why do you need them?
37. Explain the preprocessor directives
38. Distinguish between the integer arithmetic and a real arithmetic

39. Explain switch statement with suitable example
40. Explain break and continue statements
41. What is an array? Differentiate between array and an ordinary variable
42. What is a function?
43. Differentiate between the standard functions and the user-defined functions
44. What is the need for the function declaration?
45. How does the function definition differ from the function declaration?
46. Explain the parameter passing mechanisms
47. Distinguish between
    Arguments and parameters
    Local variables and global variables
48. What is a character array?
49. Explain how you initialize the character arrays
50. What is a structure?
51. Differentiate between an array and a structure
52. How do you access the member of a structure?
53. What is an embedded structure? Give an example
54. What is the need for an array of structure?
55. Explain with example
56. What is a union? How it is different from a structure?
57. What is the advantage of a union?
58. What is a pointer? Mention the advantage of pointers
59. Distinguish between an address operator and a dereferencing operator
60. How do you declare a pointer variable? Explain with suitable examples
61. What do you understand by pointer initialization? Explain with example
62. Can you pass pointers to functions? If yes, explain with a suitable example
63. What is a data file? How is it different from a structure?
64. Why data files are required?
65. How do you write data into a file?
66. Explain different type s of access modes
67. What are random access files? How do you process them?

Appendix – E

Write Programs Yourself

1. Write a C program to print your name on the computer screen
2. Write a C program to accept your name, roll number and marks in physics, chemistry, maths and computer science and print the same
3. Write a C program to find the area of triangle when its three sides are given
4. Write a C program to find the value of y in the following equation $Y = x^2 + 3x + 1$
5. Write a C program to compute the commission on sales as per the following policy

   If sales is greater than Rs. 1000 and less than Rs. 25,000- then commission is 10% of sales
   If sales is greater than Rs. 25,000 then commission is Rs. 200\-plus 8% of sales
   If sales is less than Rs. 1000 – no commission

6. Write a C program to accept a number and display it in words (EG 132 ->one three two)
7. Write a C program to accept a message and count the number of vowels and consonants in it.
8. Write a C program to illustrate the use of the break statement in a for loop.
9. Write a C program to accept the sex code of a person and display whether the person is a male or a female.
10. Find the number of positives, negatives and zeros elements in a array of 10 elements
11. Delete an element from the array
12. Insert an element in a sorted array
13. Tum of each row and each column
14. Inverse of a matrix
15. Interchange any two rows or columns in a matrix
16. Write a program to accept a matrix and determine whether it is a
    Square matrix
    Unit matrix
    Symmetric matrix
    Spare matrix (number of zeros > number of non zero elements)
17. Write a C program to compute the sum of even numbers and the sum of odd numbers using a function
18. Write a C program to compute the sum of individual digits of a given number using a function
19. Write a C program to find the maximum and minimum elements of a one-dimensional array. Use function
20. Write a C program to find the number of elements that are less than the average of a one-dimensional array. Use function.
21. Write a C program to compute the number of vowels, consonants, words and lines in a given text. Use function
22. Write a C program to accept a line of text and display the number of digits and alphabets in it.
23. Write a C program to accept a line of text and display the number of times each character appears in the text
24. Write a C program to accept a line of text and search for the word “is” and display the appropriate message
25. Write a C program to accept two strings s1 and s2 and check whether s2 is contained in s1.

26. Write a C program to accept two strings and concatenate them without using library function
27. Write a C program to accept a sentence and convert all lowercase characters to uppercase and vice-versa
28. Write a C program to accept a message and encode it by adding the value 3 to each character in the input message. Display both input and encoded messages
29. Write a C program to accept a message for a telegram and compute the bill. If there are less than 10 words then each word is charged Rs. 0.80 and if there are more than 20 words then the charge will be Rs. 10 and Rs. 0.60 for each extra word exceeding 20.
30. Write a C program to accept the name, date_of_birth, age, salary, height and employment detals of some persons and display it in an attractive manner
31. Write a C program to accept different goods with the number, price and date of purchase and display them
32. Write a C program to accept the empcode, empname, basic_salary of the employees and compute their gross salary. The gross salary is computed using the formula.
    Gross_salary = Basic_salary + DA + HRA
    Where DA – Dearness Allowance and HRA – House Rent Allowance.
33. Write a C program to read the following information of 100 students
    Student name
    Student roll number
    Marks
    The program should print the name and roll number of students who have obtained more than 60 marks.
34. Write a C program to accept two alphabets and pass them to the function via pointers which checks for the type of these alphabets. If both the alphabets are vowels then the function should return to the calling function, their previous alphabets. If both the alphabets are consonants then the function should return their successor alphabets.
35. Write a C program to find the desired element in an array of N elements. Use pointers for searching the element.
36. Write a C program to accept an array of elements and find the Largest element using the pointer
37. Write a C program to define three pointer variables for character, integer and float variables. Display the values and addresses of all the variables.
38. Write a C program to create a data file to store the information of a student in terms of
    Name
    University Register Number
    Semester
    Marks in all subjects
    Percentage
    And Display the same on the monitor
39. Write a C program to create a data file to store information such as name, age and marital status (Married / Unmarried) of a person. And read data of only unmarried persons from this file and display their names in another file.

40. Write a C program to create a data file to store information of an employee in terms of
    Name
    Employee number
    Designation and
    Salary

    Read from this file data of all employees whose salary is above Rs. 20,000 and put into another file and display it.
41. Write a C program to create a data file to store a paragraph of your choice in a file named input.txt and read the contents from this file from its end and put into a new file named output.txt

Appendix - F

Lab Exercise

1. Write a C program to find the area of a circle, given the radius

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define PI 3.142
void main()
{
        float radius, area;
        clrscr();
         printf("Enter the radius of a circle\n");
        scanf ("%f", &radius);
        area = PI * pow (radius,2);
        printf ("Area of a circle = %5.2f\n", area);
}

/*-----------------------------
```

Output

```
RUN1
Enter the radius of a circle
3.2
Area of a circle = 32.17

RUN 2
Enter the radius of a circle
6
Area of a circle = 113.11

-----------------------------*/
```

2. Write a C program to find the simple interest , given principle

```
 * rate of interest and times*/

#include <stdio.h>
#include <conio.h>
void main()
{
        float p, r, si;
        int t;
        clrscr();
        printf("Enter the values of p,r and t\n");
        scanf ("%f %f %d", &p, &r, &t);
        si = (p * r * t)/ 100.0;
```

```
        printf ("Amount = Rs. %5.2f\n", p);
        printf ("Rate   = Rs. %5.2f%\n", r);
        printf ("Time   = %d years\n", t);
        printf ("Simple interest  = %5.2f\n", si);
}

/*-----------------------------
```

Output

```
Enter the values of p,r and t
2000
8
3
Amount = Rs. 2000.00
Rate   = Rs.  8.00%
Time   = 3 years
Simple interest  = 480.00
------------------------------*/
```

3. Write a C program to check whether a given integer is odd or even

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int ival, remainder;
        clrscr();
        printf("Enter an integer :");
        scanf ("%d", &ival);
        remainder = ival % 2;
        if (remainder == 0)
        printf ("%d, is an even integer\n", ival);
        else
        printf ("%d, is an odd integer\n", ival);
}

/*-----------------------------
```

Output

```
RUN1
Enter an integer :13
13, is an odd integer

RUN2
Enter an integer :24
24, is an even integer
---------------------------------*/
```

4. Write a C program to check whether a given integer number is positive or negative

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int number;
        clrscr();
        printf("Enter a number\n");
        scanf ("%d", &number);
        if (number > 0)
        printf ("%d, is a positive number\n", number);
        else
        printf ("%d, is a negative number\n", number);
}


/*-----------------------------
Output
Enter a number
-5
-5, is a negative number

RUN2
Enter a number
89
89, is a positive number
-----------------------------*/
```

5. Write a C program to find the biggest of three numbers

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
        int  a, b, c;
        clrscr();
        printf("Enter the values of a,b and c\n");
        scanf ("%d %d %d", &a, &b, &c);
        printf ("a = %d\tb = %d\tc = %d\n", a,b,c);
        if ( a > b)
        {
                if ( a > c)
                {
                        printf ("A is the greatest among three\n");
                }
```

```
                else
                {
                        printf ("C is the greatest among three\n");
                }
        }
        else if (b > c)
        {
                printf ("B is the greatest among three\n");
        }
        else
        printf ("C is the greatest among three\n");
}

/*------------------------------
```

Output

```
Enter the values of a,b and c
23 32 45
a = 23  b = 32  c = 45
C is the greatest among three

RUN2
Enter the values of a,b and c
234
678
195
a = 234 b = 678 c = 195
B is the greatest among three

RUN3
Enter the values of a,b and c
30 20 10
a = 30  b = 20  c = 10
A is the greatest among three
------------------------------*/
```

6.Write a C program to find and output all the roots of a quadratic equation, for non-zero coefficients. In case of errors your program should report suitable error message

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
void main()
{
        float A, B, C, root1, root2;
        float realp, imagp, disc;
        clrscr();
        printf("Enter the values of A, B and C\n");
        scanf("%f %f %f", &A,&B,&C);
        /* If A = 0, it is not a quadratic equation */
        if( A==0 || B==0 || C==0)
```

```
        {
                printf("Error: Roots cannot be determined\n");
                exit(1);
        }
        else
        {
                disc = B*B - 4.0*A*C;
                if(disc < 0)
                {
                        printf("Imaginary Roots\n");
                        realp = -B/(2.0*A) ;
                        imagp = sqrt(abs(disc))/(2.0*A);
                        printf("Root1 = %f  +i %f\n",realp, imagp);
                        printf("Root2 = %f  -i %f\n",realp, imagp);
                }
                else if(disc == 0)
                {
                        printf("Roots are real and equal\n");
                        root1 = -B/(2.0*A);
                        root2 = root1;
                        printf("Root1 = %f  \n",root1);
                        printf("Root2 = %f  \n",root2);
                }
                else if(disc > 0 )
                {
                        Prinf( "Roots are real and distinct\n");
                        root1 =(-B+sqrt(disc))/(2.0*A);
                        root2 =(-B-sqrt(disc))/(2.0*A);
                        printf("Root1 = %f  \n",root1);
                        printf("Root2 = %f  \n",root2);
                }
        }
}   /* End of main() */

/*---------------------------
```

Output

```
RUN 1
Enter the values of A, B and C
3 2 1
Imaginary Roots
Root1 = -0.333333  +i 0.471405
Root2 = -0.333333  -i 0.471405

RUN 2
Enter the values of A, B and C
1 2 1
Roots are real and equal
Root1 = -1.000000
Root2 = -1.000000
```

```
RUN 3
Enter the values of A, B and C
3 5 2
Roots are real and distinct
Root1 = -0.666667
Root2 = -1.000000
---------------------------------*/
```

7. Write a C program to simulate a simple calculator to perform arithmetic operations like addition, subtraction, multiplication and division only on integers. Error message should be repoetrd if any attempt is made to divide by zero

```
#include <stdio.h>
#include <conio.h>
void main()
{
        char oper;
        /* oper is an operator to be selected */
        float n1, n2, result;
         clrscr();
        printf ("Simulation of a Simple Calculator\n\n");
        printf("Enter two numbers\n");
        scanf ("%f %f", &n1, &n2);
        fflush (stdin);
        printf("Enter the operator [+,-,*,/]\n");
        scanf ("%c", &oper);
        switch (oper)
        {
                case '+': result = n1 + n2;
                break;
                case '-': result = n1 - n2;
                break;
                case '*': result = n1 * n2;
                break;
                case '/': result = n1 / n2;
                break;
                default : printf ("Error in operation\n");
                break;
        }
        printf ("\n%5.2f %c %5.2f= %5.2f\n", n1,oper, n2, result);
}
/*----------------------------
```

Output

```
Simulation of Simple Calculator
Enter two numbers
3 5
Enter the operator [+,-,*,/]
+
 3.00 +  5.00=  8.00
```

```
RUN2
Simulation of Simple Calculator
Enter two numbers
12.75
8.45

Enter the operator [+,-,*,/]
12.75 -  8.45=  4.30

RUN3
Simulation of Simple Calculator

Enter two numbers
12 12
Enter the operator [+,-,*,/]

12.00 * 12.00= 144.00


RUN4
Simulation of Simple Calculator

Enter two numbers
5
9
Enter the operator [+,-,*,/]
/
 5.00 /  9.00=  0.56

------------------------------*/
```

8.Write a C program to find the sum of odd numbers and sum of even numbers from 1 to N. Output the computed sum on two different lines with suitable headings

```
include <stdio.h>
#include <conio.h>
void main()
{
        int i, N, oddSum = 0, evenSum = 0;
        clrscr();
        printf("Enter the value of N\n");
        scanf ("%d", &N);
        for (i=1; i <=N; i++)
        {
                 if (i % 2 == 0)
                evenSum = evenSum + i;
                else
                oddSum = oddSum + i;
```

```
        }
        printf ("Sum of all odd numbers  = %d\n", oddSum);
        printf ("Sum of all even numbers = %d\n", evenSum);
}

/*-----------------------------
```

Output

```
RUN1

Enter the value of N
10
Sum of all odd numbers  = 25
Sum of all even numbers = 30

RUN2
Enter the value of N
50
Sum of all odd numbers  = 625
Sum of all even numbers = 650

------------------------------*/
```

9. Write a C program to reverse a given integer number and check whether it is a palindrome. Output the given numbers with suitable message

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

void main()
{
        int   num, temp, digit, rev = 0;
        clrscr();
        printf("Enter an integer\n");
        scanf("%d", &num);
        temp = num;       /* original number is stored at temp */
        while(num > 0)
        {
                digit = num % 10;
                rev = rev * 10 + digit;
                num /= 10;
        }
        printf("Given number is = %d\n", temp);
        printf("Its reverse is  = %d\n", rev);
        if(temp == rev )
        printf("Number is a palindrome\n");
        else
        printf("Number is not a palindrome\n");
}
```

```
/*-------------------------
```
Output
```
RUN 1
Enter an integer
12321
Given number is = 12321
Its reverse is  = 12321
Number is a palindrome

RUN 2
Enter an integer
3456
Given number is = 3456
Its reverse is  = 6543
Number is not a palindrome
-----------------------------------*/
```

10. Write a C program to find the value of sin(x) using the series up to the given accuracy (without using user defined function) Also print sin(x) using library function.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
void main()
{
        int n, x1;
        float  acc, term, den, x, sinx=0, sinval;
        clrscr();
        printf("Enter the value of x (in degrees)\n");
        scanf("%f",&x);
        x1 = x;
        /* Converting degrees to radians*/
        x = x*(3.142/180.0);
        sinval = sin(x);
        printf("Enter the accuary for the result\n");
        scanf("%f", &acc);
        term = x;
        sinx = term;
        n = 1;
        do
        {
                den = 2*n*(2*n+1);
                term = -term * x * x / den;
                sinx = sinx + term;
                 n = n + 1;
        } while(acc <= fabs(sinval - sinx));
        printf("Sum of the sine series = %f\n", sinx);
        printf("Using Library function sin(%d) = %f\n", x1,sin(x));
}       /*End of main() */
```

```
/*------------------------------
```
Output
```
Enter the value of x (in degrees)
30
Enter the accuary for the result
0.000001
Sum of the sine series         = 0.500059
Using Library function sin(30) = 0.500059
RUN 2
Enter the value of x (in degrees)
45
Enter the accuary for the result
0.0001
Sum of the sine series         = 0.707215
Using Library function sin(45) = 0.707179
-----------------------------------------------*/
```

11. Write a C program to find the value of cos(x) using the series up to the given accuracy (without using user defined function) Also print cos(x) using library function.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
void main()
{
        int  n, x1;
        float  acc, term, den, x, cosx=0, cosval;
        clrscr();
        printf("Enter the value of x (in degrees)\n");
        scanf("%f",&x);
        x1 = x;
        /* Converting degrees to radians*/
        x = x*(3.142/180.0);
        cosval = cos(x);
        printf("Enter the accuary for the result\n");
        scanf("%f", &acc);
        term = 1;
        cosx = term;
        n = 1;
        do
        {
                den = 2*n*(2*n-1);
                term = -term * x * x / den;
                cosx = cosx + term;
                n = n + 1;
        } while(acc <= fabs(cosval - cosx));
        printf("Sum of the cosine series       = %f\n", cosx);
        printf("Using Library function cos(%d) = %f\n", x1,cos(x));
} /*End of main() */
```

```
/*------------------------------
Output
Enter the value of x (in degrees)
30
Enter the accuary for the result
0.000001
Sum of the cosine series       = 0.865991
Using Library function cos(30) = 0.865991

RUN 2
Enter the value of x (in degrees)
45
Enter the accuary for the result
0.0001
Sum of the cosine series       = 0.707031
Using Library function cos(45) = 0.707035
-----------------------------------------------*/
```

12.Write a C program to check whether a given number is prime or not and output the given number with suitable message

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
void main()
{
        int num, j, flag;
        clrscr();
        printf("Enter a number\n");
        scanf("%d", &num);
        if ( num <= 1)
        {
                printf("%d is not a prime numbers\n", num);
                exit(1);
        }
        flag = 0;
        for ( j=2; j<= num/2; j++)
        {
                if( ( num % j ) == 0)
                {
                        flag = 1;
                        break;
                }
        }
        if(flag == 0)
         printf("%d is a prime number\n",num);
        else
        printf("%d is not a prime number\n", num);
}
```

```
/*-----------------------
```
Output
```
RUN 1
Enter a number
34
34 is not a prime number

RUN 2
Enter a number
29
29 is a prime number
----------------------------*/
```

13. Write a C program to generate and print prime numbers in a given range. Also print the number of prime numbers

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
void main()
{
        int M, N, i, j, flag, temp, count = 0;
        clrscr();
        printf("Enter the value of M and N\n");
        scanf("%d %d", &M,&N);
        if(N < 2)
        {
                printf("There are no primes upto %d\n", N);
                exit(0);
        }
        printf("Prime numbers are\n");
        temp = M;
        if ( M % 2 == 0)
        {
                M++;
        }
        for (i=M; i<=N; i=i+2)
        {
                flag = 0;
                for (j=2; j<=i/2; j++)
                {
                if( (i%j) == 0)
                {
                        flag = 1;
                        break;
                }
        }
        if(flag == 0)
```

```
        {
                printf("%d\n",i);
                count++;
        }
        printf("Number of primes between %d  and %d = %d\n",temp,N,count);
}
/*---------------------------------
```

Output

```
Enter the value of M and N
15 45
Prime numbers are
17
19
23
29
31
37
41
43
Number of primes between 15  and 45 = 8
----------------------------------------*/
```

14. Write a C program to find the number of integers divisible by 5 between the given range N1 and N2, where N1 < N2 and are integers.Also find the sum of all these integer numbers that divisible by 5 and output the computed results

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int i, N1, N2, count = 0, sum = 0;
        clrscr();
        printf ("Enter the value of N1 and N2\n");
        scanf ("%d %d", &N1, &N2);
        /*Count the number and compute their sum*/
        printf ("Integers divisible by 5 are\n");
        for (i = N1; i < N2; i++)
        {
                if (i%5 == 0)
                {
                        printf("%3d,", i);
                        count++;
                        sum = sum + i;
                }
        }
        printf ("\nNumber of integers divisible by 5 between %d and %d = %d\n",  N1,N2,count);
        printf ("Sum of all integers that are divisible by 5 = %d\n", sum);

}   /* End of main()*/
```

```
/*----------------------------
```
Output
```
Enter the value of N1 and N2
2
27
Integers divisible by 5 are
  5, 10, 15, 20, 25,
Number of integers divisible by 5 between 2 and 27 = 5
Sum of all integers that are divisible by 5 = 75
-----------------------------------------------------*/
```

15. Write a C program to read N integers (zero, +ve and -ve) into an array A and to

a) Find the sum of negative numbers
b) Find the sum of positive numbers and
c) Find the average of all input numbers

Output the various results computed with proper headings */

```
#include <stdio.h>
#include <conio.h>
#define MAXSIZE 10
void main()
{
        int array[MAXSIZE];
        int i, N, negsum=0, posum=0;
        float total=0.0, averg;
        clrscr();
        printf ("Enter the value of N\n");
        scanf("%d", &N);
        printf("Enter %d numbers (-ve, +ve and zero)\n", N);
        for(i=0; i< N ; i++)
        {
                scanf("%d",&array[i]);
                fflush(stdin);
        }
        printf("Input array elements\n");
        for(i=0; i< N ; i++)
        {
                printf("%+3d\n",array[i]);
        }
        /* Summing  begins */
        for(i=0; i< N ; i++)
        {
                if(array[i] < 0)
                {
                        negsum = negsum +  array[i];
                }
                else if(array[i] > 0)
                {
```

```
                posum = posum + array[i];
            }
            else if( array[i] == 0)
            {
                ;
            }
            total = total + array[i] ;
        }
        averg = total / N;
        printf("\nSum of all negative numbers    = %d\n",negsum);
        printf("Sum of all positive numbers    = %d\n", posum);
        printf("\nAverage of all input numbers   = %.2f\n", averg);

}       /*End of main()*/

/*-------------------------------------
```

Output

```
Enter the value of N
5
Enter 5 numbers (-ve, +ve and zero)
+5
-3
0
-7
+6
Input array elements
 +5
 -3
 +0
 -7
 +6

Sum of all negative numbers    = -10
Sum of all positive numbers    = 11

Average of all input numbers   = 0.20
-------------------------------------*/
```

16.Write a C program to sort N numbers in ascending order using Bubble sort and print both the given and the sorted array with suitable headings

```
#include <stdio.h>
#include <conio.h>
#define MAXSIZE 10
void main()
{
        int array[MAXSIZE];
        int i, j, N, temp;
        clrscr();
        printf("Enter the value of N\n");
        scanf("%d",&N);
```

```
        printf("Enter the elements one by one\n");
        for(i=0; i<N ; i++)
        {
                scanf("%d",&array[i]);
        }
        printf("Input array is\n");
        for(i=0; i<N ; i++)
        {
                printf("%d\n",array[i]);
        }
        /* Bubble sorting begins */
        for(i=0; i< N ; i++)
        {
                for(j=0; j< (N-i-1) ; j++)
                {
                        if(array[j] > array[j+1])
                        {
                                temp      = array[j];
                                array[j]  = array[j+1];
                                array[j+1] = temp;
                        }
                }
        }
        printf("Sorted array is...\n");
        for(i=0; i<N ; i++)
        {
                printf("%d\n",array[i]);
        }
}  /* End of main*/

/*---------------------------------
```

Output

```
Enter the value of N
5
Enter the elements one by one
390
234
111
876
345
Input array is
390
234
111
876
345
Sorted array is...
111
234
345
```

390
876
----------*/

17. Write a C program to accept N numbers sorted in ascending order and to search for a given number using binary search. Report sucess or fialure in the form of suitable messages

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int array[10];
        int i, j, N, temp, keynum;
        int low,mid,high;
        clrscr();
        printf("Enter the value of N\n");
        scanf("%d",&N);
        printf("Enter the elements one by one\n");
        for(i=0; i<N ; i++)
        {
                scanf("%d",&array[i]);
        }
        printf("Input array elements\n");
        for(i=0; i<N ; i++)
        {
                printf("%d\n",array[i]);
        }
        /* Bubble sorting begins */
        for(i=0; i< N ; i++)
        {
                for(j=0; j< (N-i-1) ; j++)
                {
                        if(array[j] > array[j+1])
                        {
                                temp      = array[j];
                                array[j]  = array[j+1];
                                array[j+1] = temp;
                        }
                }
        }
        printf("Sorted array is...\n");
        for(i=0; i<N ; i++)
        {
                printf("%d\n",array[i]);
        }
        printf("Enter the element to be searched\n");
        scanf("%d", &keynum);
        /* Binary searching begins */
        low=1;
        high=N;
```

```
        do
        {
                mid= (low + high) / 2;
                if ( keynum < array[mid] )
                high = mid - 1;
                else if ( keynum > array[mid])
                low = mid + 1;
        } while( keynum!=array[mid] && low <= high); /* End of do- while */
        if( keynum == array[mid] )
        {
                printf("SUCCESSFUL SEARCH\n");
        }
        else
        {
                printf("Search is FAILED\n");
        }

}  /* End of main*/

/*---------------------------------
Output
Enter the value of N
4
Enter the elements one by one
3
1
4
2
Input array elements
3
1
4
2
Sorted array is...
1
2
3
4
Enter the element to be searched
4
SUCCESSFUL SEARCH
--------------------------*/
```

18. Write a C program to input real numbers and find the mean, variance and standard deviation

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define MAXSIZE 10
```

```
void main()
{
        float x[MAXSIZE];
        int  i, n;
        float avrg, var, SD, sum=0, sum1=0;
        clrscr();
        printf("Enter the value of N\n");
        scanf("%d", &n);
        printf("Enter %d real numbers\n",n);
        for(i=0; i<n; i++)
        {
                scanf("%f", &x[i]);
        }
        /* Compute the sum of all elements */
        for(i=0; i<n; i++)
        {
                sum = sum + x[i];
        }
        avrg = sum /(float) n;
        /* Compute  varaience  and standard deviation  */
        for(i=0; i<n; i++)
        {
                sum1 = sum1 + pow((x[i] - avrg),2);
        }
        var = sum1 / (float) n;
        SD = sqrt(var);
        printf("Average of all elements  = %.2f\n", avrg);
        printf("Varience of all elements = %.2f\n", var);
        printf("Standard deviation       = %.2f\n", SD);
}  /*End of main()*/

/*--------------------------
```

Output

```
Enter the value of N
6
Enter 6 real numbers
12
34
10
50
42
33
Average of all elements  = 29.66
Varience of all elements = 213.89
Standard deviation       = 14.62
-------------------------------------*/
```

19. Write a C program to read in four integer numbers into an array and find the average of largest two of the given numbers without sorting the array. The program should output the given four numbers and the average with suitable headings.

```
#include <stdio.h>
#include <conio.h>
#define MAX 4
void main()
{
        int a[MAX], i, l1,l2,temp;
        clrscr();
        printf("Enter %d integer numbers\n", MAX);
        for (i=0; i < MAX; i++)
        {
                scanf("%d", &a[i]);
        }
        printf("Input interger are\n");
        for (i=0; i < MAX; i++)
        {
                printf("%5d", a[i]);
        }
        printf("\n");
        l1 = a[0];  /*assume first element of array is the first largest*/
        l2 = a[1];  /*assume first element of array is the second largest*/
        if (l1 < l2)
        {
                temp = l1;
                l1 = l2;
                l2 = temp;
        }
        for (i=2;i<4;i++)
        {
                if (a[i] >= l1)
                {
                        l2 = l1;
                        l1 = a[i];
                }
                else if(a[i] > l2)
                {
                        l2= a[i];
                }
        }
        printf("\n%d is the first largest\n", l1);
        printf("%d is the second largest\n", l2);
        printf("\nAverage of %d and %d = %d\n", l1,l2, (l1+l2)/2);
}
```

```
/*-----------------------------------
```
Output
```
RUN 1
Enter 4 integer numbers
45
33
21
10
Input interger are
  45   33   21   10

45 is the first largest
33 is the second largest

Average of 45 and 33 = 39

RUN 2
Enter 4 integer numbers
12
90
54
67
Input interger are
  12   90   54   67

90 is the first largest
67 is the second largest

Average of 90 and 67 = 78

RUN 3
Enter 4 integer numbers
100
200
300
400
Input interger are
 100  200  300  400

400 is the first largest
300 is the second largest

Average of 400 and 300 = 350

-----------------------------------*/
```

20. Write a C program to evaluate the given polynomial P(x)=AnXn + An-1Xn-1 + An-2Xn-2+... +A1X + A0, by reading its coefficents into an array. [Hint:Rewrite the polynomial as P(x) = a0 + x(a1+x(a2+x(a3+x(a4+x(...x(an-1+xan)))) and evalate the function starting from the inner loop]/

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define MAXSIZE 10
voidmain()
{
        int a[MAXSIZE];
        int i, N,power;
        float x, polySum;
        clrscr();
        printf("Enter the order of the polynomial\n");
        scanf("%d", &N);
        printf("Enter the value of x\n");
        scanf("%f", &x);
        /*Read the coefficients into an array*/
        printf("Enter %d coefficients\n",N+1);
        for (i=0;i <= N;i++)
        {
                scanf("%d",&a[i]);
        }
        polySum = a[0];
        for (i=1;i<= N;i++)
        {
                polySum = polySum * x + a[i];
        }
        power = N;
        /*power--;*/
        printf("Given polynomial is:\n");
        for (i=0;i<= N;i++)
        {
                if (power < 0)
                {
                        break;
                }
                /* printing proper polynomial function*/
                if (a[i] > 0)
                printf(" + ");
                else if (a[i] < 0)
                printf(" - ");
                else
                printf (" ");
                printf("%dx^%d  ",abs(a[i]),power--);
        }
        printf("\nSum of the polynomial = %6.2f\n",polySum);
}
```

```
/*-------------------------------------------------------
```

Output

```
RUN 1
Enter the order of the polynomial
2
Enter the value of x
2
Enter 3 coefficients
3
2
6
Given polynomial is:
 + 3x^2   + 2x^1   + 6x^0
Sum of the polynomial =  22.00

RUN 2
Enter the order of the polynomial
4
Enter the value of x
1
Enter 5 coefficients
3
-5
6
8
-9

Given polynomial is:
 + 3x^4   - 5x^3   + 6x^2   + 8x^1   - 9x^0
Sum of the polynomial =   3.00
-------------------------------------------------------*/
```

21. Write a C program to read two matrices A (MxN) and B(MxN) and perform addition OR subtraction of A and B. Find the trace of the resultant matrix. Output the given matrix, their sum or Differences and the trace

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int A[10][10], B[10][10], sumat[10][10], diffmat[10][10];
        int i, j, M,N, option;
        void trace (int arr[][10], int M, int N);
        clrscr();
        printf("Enter the order of the matrice A and B\n");
        scanf("%d %d", &M, &N);
        printf("Enter the elements of matrix A\n");
        for(i=0; i<M; i++)
        {
```

```
        for(j=0; j<N; j++)
        {
                scanf("%d",&A[i][j]);
        }
}
printf("MATRIX A is\n");
for(i=0; i<M; i++)
{
        for(j=0; j<N; j++)
        {
                printf("%3d",A[i][j]);
        }
        printf("\n");
}
printf("Enter the elements of matrix B\n");
for(i=0; i<M; i++)
{
        for(j=0; j<N; j++)
        {
                scanf("%d",&B[i][j]);
        }
}
printf("MATRIX B is\n");
for(i=0; i<M; i++)
{
        for(j=0; j<N; j++)
        {
                printf("%3d",B[i][j]);
        }
        printf("\n");
}
printf("Enter your option: 1 for Addition and 2 for Subtraction\n");
scanf("%d",&option);
switch (option)
{
        case 1: for(i=0; i<M; i++)
        {
                for(j=0; j<N; j++)
                {
                        sumat[i][j] = A[i][j] + B[i][j];
                }
        }
        printf("Sum matrix is\n");
        for(i=0; i<M; i++)
        {
                for(j=0; j<N; j++)
                {
                        printf("%3d",sumat[i][j]) ;
                }
                printf("\n");
```

```
                }
                trace (sumat, M, N);
                break;
                case 2:for(i=0; i<M; i++)
                {
                        for(j=0; j<N; j++)
                        {
                                diffmat[i][j] = A[i][j] - B[i][j];
                        }
                }
                printf("Difference matrix is\n");
                for(i=0; i<M; i++)
                {
                        for(j=0; j<N; j++)
                        {
                                printf("%3d",diffmat[i][j]) ;
                        }
                        printf("\n");
                }
                trace (diffmat, M, N);
                break;
        }

}    /* End of main() */
/*Function to find the trace of a given matrix and print it*/
void trace (int arr[][10], int M, int N)
{
        int i, j, trace = 0;
        for(i=0; i<M; i++)
        {
                for(j=0; j<N; j++)
                {
                        if (i==j)
                        {
                                trace = trace + arr[i][j];
                        }
                }
        }
        printf ("Trace of the resultant matrix is = %d\n", trace);

}

/*----------------------------------
```

Output

```
Enter the order of the matrice A and B
2 2
Enter the elements of matrix A
1 1
2 2
MATRIX A is
```

```
 1  1
 2  2
Enter the elements of matrix B
3 3
4 4
MATRIX B is
 3  3
 4  4

Enter your option: 1 for Addition and 2 for Subtraction
1
Sum matrix is
 4  4
 6  6
Trace of the resultant matrix is = 10
-----------------------------------------------*/
```

22. Write a C program to read A (MxN), find the transpose of a given matrix and output both the input matrix and the tansposed matrix/

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int i,j,M,N;
        int A[10][10], B[10][10];
        int transpose(int A[][10], int r, int c);  /*Function prototype*/
        clrscr();
        printf("Enter the order of matrix A\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("Matrix A is\n");
        for(i=0;i<M;i++)
        {
                for(j=0;j<N;j++)
                {
                        printf("%3d",A[i][j]);
                }
                printf("\n");
        }
        /* Finding Transpose of matrix*/
        for(i=0;i<M;i++)
        {
```

```
                for(j=0;j<N;j++)
                {
                        B[i][j] =  A[j][i];
                }
        }
        printf("Its Transpose is\n");
        for(i=0;i<M;i++)
        {
                for(j=0;j<N;j++)
                {
                        printf("%3d",B[i][j]);
                }
                printf("\n");
        }

}       /*End of main()*/

/*---------------------------------------
```

Output

```
Enter the order of matrix A
3 3
Enter the elements of matrix
1
2
3
4
5
6
7
8
9
MatrixxA is
  1  2  3
  4  5  6
  7  8  9
Its Transpose is
  1  4  7
  2  5  8
  3  6  9
-----------------------------*/
```

23. Write a C program to read a string and check whether it is a palindrome or not (without using library functions). Output the given string along with suitable message

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
        char string[25], revString[25]={'\0'};
```

```
        int  i,length = 0, flag = 0;
        clrscr();
        fflush(stdin);
        printf("Enter a string\n");
        gets(string);
        for (i=0; string[i] != '\0'; i++)
        {
                /*keep going through each */
                /*character of the string */
                length++; /*till its end */
        }
        for (i=length-1; i >= 0 ; i--)
        {
                revString[length-i-1] = string[i];
        }
        /*Compare the input string and its reverse. If both are equal then the input string is
        palindrome. Otherwise it is not a palindrome */
        for (i=0; i < length ; i++)
        {
                if (revString[i] == string[i])
                flag = 1;
                else
                flag = 0;
        }
        if (flag == 1)
        printf ("%s is a palindrome\n", string);
        else
        printf("%s is not a palindrome\n", string);

}   /*End of main()*/

/*--------------------------------------------------
```

Output

```
RUN 1
Enter a string
madam
madam is a palindrome

RUN 2
Enter a string
Madam
Madam is not a palindrome

RUN 3
Enter a string
good
good is not a palindrome
-----------------------------------------------------------*/
```

24.Write a C program to read an English sentence and replace lowercase characters by uppercase and vice-versa. Output the given sentence as well as the case covrted sentence on two different lines.

```
#include <stdio.h>
#include <ctype.h>
#include <conio.h>
void main()
{
        char sentence[100];
        int count, ch, i;

        clrscr();
        printf("Enter a sentence\n");
        for(i=0; (sentence[i] = getchar())!='\n'; i++)
        {
                ;
        }
        sentence[i]='\0';
        count = i; /*shows the number of chars accepted in a sentence*/
        printf("The given sentence is   : %s",sentence);
        printf("\nCase changed sentence is: ");
        for(i=0; i < count; i++)
        {
                ch = islower(sentence[i]) ? toupper(sentence[i]) : tolower(sentence[i]);
                putchar(ch);
        }
} /*End of main()*/

/*------------------------------
Output
Enter a sentence
Mera Bharat Mahan
The given sentence is   : Mera Bhaaat Mahan
Case changed sentence is: mERA bHARAT mAHAN
-----------------------------------------------*/
```

25. Write a C program read a sentence and count the number of vowels and consonants in the given sentence. Output the results on two lines with suitable headings

```
#include <stdio.h>
#include <conio.h>
void main()
{
        char sentence[80];
        int i, vowels=0, consonants=0, special = 0;
        clrscr();
        printf("Enter a sentence\n");
        gets(sentence);
```

```
        for(i=0; sentence[i] != '\0'; i++)
        {
                if((sentence[i] == 'a'||sentence[i] == 'e'||sentence[i] == 'i'||
                sentence[i] == 'o'||sentence[i] == 'u') ||(sentence[i] == 'A'||
                sentence[i] == 'E'||sentence[i] == 'I'|| sentence[i] == 'O'||
                sentence[i] == 'U'))
                {
                        vowels = vowels + 1;
                }
                else
                {
                        consonants = consonants + 1;
                }
                if (sentence[i] =='\t' ||sentence[i] =='\0' || sentence[i] ==' ')
                {
                        special = special + 1;
                }
        }
        consonants = consonants - special;
        printf("No. of vowels in %s = %d\n", sentence, vowels);
        printf("No. of consonants in %s = %d\n", sentence, consonants);
}

/*---------------------------------------
```

Output

```
Enter a sentence
Good Morning
No. of vowels in Good Morning = 4
No. of consonants in Good Morning = 7
---------------------------------------*/
```

26. Write a C program to read N names, store them in the form of an array and sort them in alphabetical order. Output the give names and the sorted names in two columns side by side wih suitable heading

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
        char name[10][8], Tname[10][8], temp[8];
        int i, j, N;
        clrscr();
        printf("Enter the value of N\n");
        scanf("%d", &N);
        printf("Enter %d names\n", N);
        for(i=0; i< N ; i++)
        {
                scanf("%s",name[i]);
                strcpy (Tname[i], name[i]);
```

```
        }
        for(i=0; i < N-1 ; i++)
        {
                for(j=i+1; j< N; j++)
                {
                        if(strcmpi(name[i],name[j]) > 0)
                        {
                                strcpy(temp,name[i]);
                                strcpy(name[i],name[j]);
                                strcpy(name[j],temp);
                        }
                }
        }
        printf("\n----------------------------------------\n");
        printf("Input Names\tSorted names\n");
        printf("----------------------------------------\n");
        for(i=0; i< N ; i++)
        {
                printf("%s\t\t%s\n",Tname[i], name[i]);
        }
        printf("----------------------------------------\n");
}/* End of main() */

/*--------------------------------
```

Output

```
Enter the value of N
3
Enter 3 names
Monica
Laxmi
Anand
---------------------------------------
Input Names     Sorted names
---------------------------------------
Monica          Anand
Laxmi           Laxmi
Anand           Monica
\*---------------------------------------   */
```

27 Write a C program to sort given N elements using SELECTION sort method using functions

a) To find maximum of elements
b) To swap two elements/

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

```
void main()
{
        int array[10];
        int i, j, N, temp;
        int findmax(int b[10], int k);
        /* function declaration */
        void exchang(int b[10], int k);
        clrscr();
        printf("Enter the value of N\n");
        scanf("%d",&N);
        printf("Enter the elements one by one\n");
        for(i=0; i<N ; i++)
        {
                scanf("%d",&array[i]);
        }
        printf("Input array elements\n");
        for(i=0; i<N ; i++)
        {
                printf("%d\n",array[i]);
        }
        /* Selection sorting begins */
         exchang(array,N);
        printf("Sorted array is...\n");
        for(i=0; i< N ; i++)
        {
                printf("%d\n",array[i]);
        }

}               /* End of main*/
/* function to find the maximum value */
int findmax(int b[10], int k)
{
        int max=0,j;
        for(j = 1; j <= k; j++)
        {
                if ( b[j] > b[max])
                {
                        max = j;
                }
        }
        return(max);
 }
void exchang(int b[10],int k)
 {
        int  temp, big, j;
        for ( j=k-1; j>=1; j--)
        {
                big = findmax(b,j);
                temp = b[big];
```

```
                b[big] = b[j];
                b[j] = temp;
        }
        return;
}

/*----------------------------
```

Output

```
Enter the value of N
5
Enter the elements one by one
45
12
90
33
78
Input array elements
45
12
90
33
78
Sorted array is
12
33
45
78
90
-----------------------------------*/
```

28. Develop functions
a) To read a given matrix
b) To output a matrix
c) To compute the product of twomatrices

Use the above functions to read in two matrices A (MxN)*
B (NxM), to compute the product of the two matrices, to output the given matrices and the computed matrix in a main function

```
#include <stdio.h>
#include <conio.h>
#define MAXROWS 10
#define MAXCOLS 10
void main()
{
        int A[MAXROWS][MAXCOLS], B[MAXROWS][MAXCOLS], C[MAXROWS][MAXCOLS];
        int M, N;
        /*Function declarations*/
        void readMatrix(int arr[][MAXCOLS], int M, int N);
        void printMatrix(int arr[][MAXCOLS], int M, int N);
        void productMatrix(int A[][MAXCOLS], int B[][MAXCOLS], int C[][MAXCOLS],int M, int N);
```

```
	clrscr();
	printf("Enter the value of M and N\n");
	scanf("%d %d",&M, &N);
	printf ("Enter matrix A\n");
	readMatrix(A,M,N);
	printf("Matrix A\n");
	printMatrix(A,M,N);
	printf ("Enter matrix B\n");
	readMatrix(B,M,N);
	printf("Matrix B\n");
	printMatrix(B,M,N);
	productMatrix(A,B,C, M,N);
	printf ("The product matrix is\n");
	printMatrix(C,M,N);
}
/*Input matrix A*/
void readMatrix(int arr[][MAXCOLS], int M, int N)
{
	int i, j;
	for(i=0; i< M ; i++)
	{
		for ( j=0; j < N; j++)
		{
			scanf("%d",&arr[i][j]);
		}
	}
}
void printMatrix(int arr[][MAXCOLS], int M, int N)
{
	int i, j;
	for(i=0; i< M ; i++)
	{
		for ( j=0; j < N; j++)
		{
			printf("%3d",arr[i][j]);
		}
		printf("\n");
	}
}
/* Multiplication of matrices */
void productMatrix(int A[][MAXCOLS], int B[][MAXCOLS], int C[][MAXCOLS],int M, int N)
{
	int i, j, k;
	for(i=0; i< M ; i++)
	{
		for ( j=0; j < N; j++)
		{
			C[i][j] = 0 ;
			for (k=0; k < N; k++)
			{
				C[i][j] = C[i][j] + A[i][k] * B[k][j];
			}
		}
	}
}
```

```
/*-----------------------------------------
```
Output
```
Enter the value of M and N
3 3
Enter matrix A
1 1 1
2 2 2
3 3 3
Matrix A
1  1  1
2  2  2
3  3  3
Enter matrix B
1 2 3
4 5 6
7 8 9
Matrix B
1  2  3
4  5  6
7  8  9
The product matrix is
12 15 18
24 30 36
36 45 54
---------------------------------*/
```

29. Write a C program to read a matrix A (MxN) and to find the following using functions
a) Sum of the elements of each row b) Sum of the elements of each column
c) Find the sum of all the elements of the matrix Output the computed results with suitable headings /

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int arr[10][10];
        int i, j, row, col, rowsum, colsum,sumall=0;
        /* Function declaration */
        int Addrow(int A[10][10], int k, int c);
        int Addcol(int A[10][10], int k, int c);
        clrscr();
        printf("Enter the order of the matrix\n");
        scanf("%d %d", &row, &col);
        printf("Enter the elements of the matrix\n");
        for(i=0; i<row; i++)
        {
                for(j=0; j< col; j++)
                {
                        scanf("%d", &arr[i][j]);
```

```
                }
        }
        printf("Input matrix is\n");
        for(i=0; i<row; i++)
        {
                for(j=0;j<col;j++)
                {
                        printf("%3d", arr[i][j]);
                }
                printf("\n");
        }
        /* computing row sum */
        for(i=0; i<row; i++)
        {
                rowsum = Addrow(arr,i,col);
                printf("Sum of row %d = %d\n", i+1, rowsum);
        }
        for(j=0; j<col; j++)
        {
                colsum = Addcol(arr,j,row);
                printf("Sum of column  %d = %d\n", j+1, colsum);
        }
        /* computation of all elements */
        for(j=0; j< row; j++)
        {
                sumall = sumall + Addrow(arr,j,col);
        }
        printf("Sum of all elements of matrix = %d\n", sumall);
        }
        /* Function to add each row */
        int Addrow(int A[10][10], int k, int c)
        {
        int rsum=0, i;
        for(i=0; i< c; i++)
        {
                rsum = rsum + A[k][i];
        }
        return(rsum);
}
/* Function to add each column */
int Addcol(int A[10][10], int k, int r)
{
        int csum=0, j;
        for(j=0; j< r; j++)
        {
                csum = csum + A[j][k];
        }
        return(csum);
}
```

```
/*----------------------------
```
Output
```
Enter the order of the matrix
3
3
Enter the elements of the matrix
1 2 3
4 5 6
7 8 9
Input matrix is
  1  2  3
  4  5  6
  7  8  9
Sum of row 1 = 6
Sum of row 2 = 15
Sum of row 3 = 24
Sum of column  1 = 12
Sum of column  2 = 15
Sum of column  3 = 18
Sum of all elements of matrix = 45

---------------------------------------*/
```

30. Write a C program to read two integers M and N and to swap their values. Use a user-defined function for swapping. Output the values of M and N before and after swapping with suitable mesages

```
#include <stdio.h>
#include <conio.h>
void main()
{
        float M,N;
        void swap(float *ptr1, float  *ptr2 );
        /* Function Declaration */
        printf("Enter the values of M and N\n");
        scanf("%f %f", &M, &N);
        printf ("Before Swapping:M = %5.2f\tN = %5.2f\n", M,N);
        swap(&M, &N);
        printf ("After Swapping:M  = %5.2f\tN = %5.2f\n", M,N);

}    /* End of main() */
/* Function swap - to interchanges teh contents of two items*/
void swap(float *ptr1, float *ptr2 )
{
        float temp;
        temp=*ptr1;
         *ptr1=*ptr2;
        *ptr2=temp;
} /* End of Function */
```

```
/* ---------------------------------------
```
Output
```
Enter the values of M and N
32 29
Before Swapping:M = 32.00       N = 29.00
After Swapping:M  = 29.00       N = 32.00
-----------------------------------------*/
```

31. Write a C program to convert the given binary number into decimal

```
#include <stdio.h>
void main()
{
        int   num, bnum, dec = 0, base = 1, rem ;
        printf("Enter a binary number(1s and 0s)\n");
        scanf("%d", &num);
        /*maximum five digits */
        bnum = num;
        while( num > 0)
        {
                 rem = num % 10;
                dec = dec + rem * base;
                num = num / 10 ;
                base = base * 2;
        }
        printf("The Binary number is = %d\n", bnum);
        printf("Its decimal equivalent is =%d\n", dec);
}

/*-------------------------------------------
```
Output
```
Enter a binary number(1s and 0s)
10101
The Binary number is = 10101
Its decimal equivalent is =21
-------------------------------------------*/
```

32. Program to accept N integer number and store them in an array AR. The odd elements in the AR are copied into OAR and other elements are copied into EAR. Display the contents of OAR and EAR

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

```
        long int ARR[10], OAR[10], EAR[10];
        int i,j=0,k=0,n;
        printf("Enter the size of array AR\n");
        scanf("%d",&n);
        printf("Enter the elements of the array\n");
        for(i=0;i<n;i++)
        {
                scanf("%ld",&ARR[i]);
                fflush(stdin);
        }
        /*Copy odd and even elemets into their respective arrays*/
        for(i=0;i<n;i++)
        {
                if (ARR[i]%2 == 0)
                {
                        EAR[j] = ARR[i];
                        j++;
                }
                else
                {
                        OAR[k] = ARR[i];
                        k++;
                }
        }
        printf("The elements of OAR are\n");
        for(i=0;i<j;i++)
        {
                printf("%ld\n",OAR[i]);
        }
        printf("The elements of EAR are\n");
        for(i=0;i<k;i++)
        {
                printf("%ld\n", EAR[i]);
        }
}       /*End of main()*/

/*-------------------------------------
```

Output

```
Enter the size of array AR
6
Enter the elements of the array
12
345
678
```

```
899
900
111
The elements of OAR are
345
899
111
The elements of EAR are
12
678
900
---------------------------------------*/
```

33.Write a C program to generate Fibonacci sequence Fibonacci sequence is 0 1 1 2 3 5 8 13 21 ...

```
#include <stdio.h>
void main()
{
        int   fib1=0, fib2=1, fib3, limit, count=0;
        printf("Enter the limit to generate the fibonacci sequence\n");
        scanf("%d", &limit);
        printf("Fibonacci sequence is ...\n");
        printf("%d\n",fib1);
        printf("%d\n",fib2);
        count = 2;
        /* fib1 and fib2 are already used */
        while( count < limit)
        {
                fib3 = fib1 + fib2;
                count ++;
                printf("%d\n",fib3);
                fib1 = fib2;
                fib2 = fib3;
        }
}
 /*-------------------------------------------------------
Enter the limit to generate the fibonacci sequence
7
Fibonacci sequence is ...
0
1
1
2
```

```
3
5
8
------------------------------------------------------------*/
```

34. Write a C program to insert a particular element in a specified position in a given array

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int  x[10];
        int  i, j, n, m, temp, key, pos;
        clrscr();
        printf("Enter how many elements\n");
        scanf("%d", &n);
        printf("Enter the elements\n");
        for(i=0; i<n; i++)
        {
                scanf("%d", &x[i]);
        }
        printf("Input array elements are\n");
        for(i=0; i<n; i++)
        {
                printf("%d\n", x[i]);
        }
        for(i=0; i< n; i++)
        {
                for(j=i+1; j<n; j++)
                {
                        if (x[i] > x[j])
                        {
                                temp = x[i];
                                x[i] = x[j];
                                x[j] = temp;
                        }
                }
        }
        printf("Sorted list is\n");
        for(i=0; i<n; i++)
        {
                 printf("%d\n", x[i]);
        }
```

```
        printf("Enter the element to be inserted\n");
        scanf("%d",&key);
        for(i=0; i<n; i++)
        {
                if ( key < x[i] )
                {
                        pos = i;
                        break;
                }
        }
        m = n - pos + 1 ;
        for(i=0; i<= m ; i++)
        {
                 x[n-i+2] = x[n-i+1] ;
        }
        x[pos] = key;
        printf("Final list is\n");
        for(i=0; i<n+1; i++)
        {
                printf("%d\n", x[i]);
        }
}

/*-------------------------------------
```

Output

```
Enter how many elements
5
Enter the elements
2
14
67
83
29
Input array elements are
2
14
67
83
29
Sorted list is
2
14
29
67
```

```
83
Enter the element to be inserted
34

Final list is
2
14
29
34
67
83
-------------------------------------------------*/
```

35. Write a C program to accept an integer and reverse it

```
#include <stdio.h>
void main()
{
        long  num, rev = 0, temp, digit;
        printf("Enter the number\n");  /*For better programming,choose 'long int' */
        scanf("%ld", &num);
        temp = num;
        while(num > 0)
        {
                digit = num % 10;
                rev = rev * 10 + digit;
                num /= 10;
        }
        printf("Given number   = %ld\n", temp);
        printf("Its reverse is = %ld\n", rev);
}

/* ---------------------------
```

Output

```
Enter the number
123456
Given number   = 123456
Its reverse is = 654321
-------------------------------*/
```

36. This program is to illustrate how user authentication is made before allowing the user to access the secured resources. It asks for the user name and then the password. The password that you enter will not be displayed, instead that character is replaced by

```
#include <stdio.h>
#include <conio.h>
void main()
{
        char pasword[10],usrname[10], ch;
        int i;
        clrscr();
        printf("Enter User name: ");
        gets(usrname);
        printf("Enter the password <any 8 characters>: ");
        for(i=0;i<8;i++)
        {
                ch = getch();
                pasword[i] = ch;
                ch = '*' ;
                printf("%c",ch);
        }
        pasword[i] = '\0';
        /*If you want to know what you have entered as password, you can print it*/
        printf("\nYour password is :");
        for(i=0;i<8;i++)
        {
                printf("%c",pasword[i]);
        }
}

/*-----------------------------------------------
```

Output

```
Enter User name: Latha
Enter the password <any 8 characters>: ********
Your password is :Wipro123
-----------------------------------------------*/
```

37.Write a C program to accept a string and a substring and check if the substring is present in the given strig

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char str[80],search[10];
        int count1=0,count2=0,i,j,flag;
        clrscr();
        puts("Enter a string:");
```

```
        gets(str);
        puts("Enter search substring:");
        gets(search);
        while (str[count1]!='\0')
        count1++;
        while (search[count2]!='\0')
        count2++;
        for(i=0;i<=count1-count2;i++)
        {
                for(j=i;j<i+count2;j++)
                {
                        flag=1;
                        if (str[j]!=search[j-i])
                        {
                                flag=0;
                                break;
                        }
                }
                if (flag==1)
                break;
        }
        if (flag==1)
        puts("SEARCH SUCCESSFUL!");
        else
        puts("SEARCH UNSUCCESSFUL!");
        getch();
}
/*-----------------------------
```

Output

```
Enter a string:
Hello how are you?
Enter search substring:
how
SEARCH SUCCESSFUL!
-----------------------------*/
```

38. Write a c program to compute the surface area and volume of a cube

```
 #include <stdio.h>
 #include <math.h>
void main()
{
        float  side, surfArea, volume;
```

```
        printf("Enter the length of a side\n");
        scanf("%f", &side);
        surfArea = 6.0 * side * side;
        volume = pow (side, 3);
        printf("Surface area = %6.2f and Volume = %6.2f\n", surfArea, volume);
}

/*----------------------------------------
```

Output

```
Enter the llngth of a side
4
Surface area =  96.00 and Volume =  64.00

RUN2
Enter the length of a side
12.5
Surface area = 937.50 and Volume = 1953.12
----------------------------------------*/
```

39. Write a c program to find whether a given year is leap year or not

```
#include <stdio.h>
void main()
{
        int year;
        printf("Enter a year\n");
        scanf("%d",&year);
        if ( (year % 4) == 0)
        printf("%d is a leap year",year);
        else
        printf("%d is not a leap year\n",year);
}

/*----------------------------------------
```

Output

```
Enter a year
2000
2000 is a  leap year

RUN2
Enter a year
2007
2007 is not a leap year
----------------------------------------*/
```

40. Write a c program to swap the contents of two numbers using bitwise XOR operation. Don't use either the temporary variable or arithmetic operators

```
 #include <stdio.h>
void main()
 {
        long i,k;
        printf("Enter two integers\n");
        scanf("%ld %ld",&i,&k);
        printf("\nBefore swapping i= %ld and k = %ld",i,k);
        i = i^k;
        k = i^k;
        i = i^k;
        printf("\nAfter swapping i= %ld and k = %ld",i,k);
}

/*----------------------------------------
Output
Enter two integers
23 34

Before swapping i= 23 and k = 34
After swapping i= 34 and k = 23
----------------------------------------*/
```

41. Write a c program to convert given number of days to a measure of time given in years, weeks and days. For example 375 days is equal to 1 year 1 week and 3 days (ignore leap year)

```
 #include <stdio.h>
 #define DAYSINWEEK 7
void main()
 {
        int ndays, year, week, days;
        printf("Enter the number of days\n");
        scanf("%d",&ndays);
        year = ndays/365;
        week = (ndays % 365)/DAYSINWEEK;
        days = (ndays%365) % DAYSINWEEK;
        printf ("%d is equivalent to %d years, %d weeks and %d days\n", ndays, year, week, days);
 }
```

```
/*---------------------------------------------
```

Output

```
Enter the number of days
375
375 is equivalent to 1 years, 1 weeks and 3 days

Enter the number of dayy
423
423 is equivalent tt 1 years, 8 weeks and 2 days

Enter the number of days
1497
1497 is equivalent to 4 years, 5 weeks and 2 days
-----------------------------------------------------*/
```

42. Write a c program to convert given number of days to a measure of time given in years, weeks and days. For example 375 days is equal to 1 year 1 week and 3 days (ignore leap year)

```
#include <stdio.h>
#define DAYSINWEEK 7
void main()
{
        int ndays, year, week, days;
        printf("Enter the number of days\n");
        scanf("%d",&ndays);
        year = ndays/365;
        week = (ndays % 365)/DAYSINWEEK;
        days = (ndays%365) % DAYSINWEEK;
        printf ("%d is equivalent to %d years, %d weeks and %d days\n",ndays, year, week, days);
}
```

```
/*---------------------------------------------
```

Output

```
Enter the number of days
375
375 is equivalent to 1 years, 1 weeks and 3 days

Enter the number of dayy
423
423 is equivalent tt 1 years, 8 weeks and 2 days
```

```
Enter the number of days
1497
1497 is equivalent to 4 years, 5 weeks and 2 days
-----------------------------------------------------*/
```

43. Program to accept two strings and concatenate them, i.e.The second string is appended to the end of the first string

```
#include <stdio.h>
#include <string.h>
void main()
{
        char string1[20], string2[20];
        int i,j,pos;
        strset(string1, '\0'); /*set all occurrences in two strings to NULL*/
        strset(string2,'\0');
        printf("Enter the first string :");
        gets(string1);
        fflush(stdin);
        printf("Enter the second string:");
        gets(string2);
        printf("First string  = %s\n", string1);
        printf("Second string = %s\n", string2);
        /*To concate the second stribg to the end of the string
        travserse the first to its end and attach the second string*/
        for (i=0; string1[i] != '\0'; i++)
        {
                ; /*null statement: simply trvsering the string1*/
        }
        pos = i;
        for (i=pos,j=0; string2[j]!='\0'; i++)
        {
                string1[i] = string2[j++];
        }
        string1[i]='\0';   /*set the last character of string1 to NULL*/
        printf("Concatenated string = %s\n", string1);
}

/*---------------------------------------
```

Output

```
Enter the first string :CD-
Enter the second string:ROM
First string  = CD-
```

```
Second string = ROM
Concatenated string = CD-ROM
---------------------------------------*/
```

44. Write a c program to find the length of a string w ithout using the built-in function

```
#include <stdio.h>
void main()
{
        char string[50];
        int  i, length = 0;
        printf("Enter a string\n");
        gets(string);
        for (i=0; string[i] != '\0'; i++) /*keep going through each */
        {
                /*character of the string */
                length++;
                /*till its end */
        }
        printf("The length of a string is the number of characters in it\n");
        printf("So, the length of %s =%d\n", string, length);
}

/*-------------------------------------------------
```

Output

```
Enter a string
hello
The length of a string is the number of characters in it
So, the length of hello = 5

RUN2
Enter a string
E-Commerce is hot now
The length of a string is the number of characters in it
So, the length of E-Commerce is hot now =21
-----------------------------------------------------------*/
```

45. Write a c program to find the length of a string without using the built-in function also check whether it is a palindrome or not

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

```
{
        char string[25], revString[25]={'\0'};
        int  i,length = 0, flag = 0;
        clrscr();
        fflush(stdin);
        printf("Enter a string\n");
        gets(string);
        for (i=0; string[i] != '\0'; i++) /*keep going through each */
        {
                /*character of the string */
                length++;
                /*till its end */
        }
        printf("The length of the string \'%s\' = %d\n", string, length);
        for (i=length-1; i >= 0 ; i--)
        {
                revString[length-i-1] = string[i];
        }
        /*Compare the input string and its reverse. If both are equal
        then the input string is palindrome. Otherwise it is not a palindrome */
        for (i=0; i < length ; i++)
        {
                if (revString[i] == string[i])
                flag = 1;
                else
                flag = 0;

        }
        if (flag == 1)
        printf ("%s is a palindrome\n", string);
        else
        printf("%s is not a palindrome\n", string);
}

/*-------------------------------------------------
```

Output

```
Enter a string
madam
The length of the string 'madam' = 5
madam is a palindrome

RUN2
Enter a string
good
```

```
The length of the string 'good' = 4
good is not a palindrome
------------------------------------------------------------*/
```

46. Write a C program to accept a string and find the number of times the word 'the' appears in it

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int count=0,i,times=0,t,h,e,space;
        char str[100];
        clrscr();
        puts("Enter a string:");
        gets(str);
        /*Traverse the string to count the number of characters*/
        while (str[count]!='\0')
        {
                count++;
        }
        /*Finding the frequency of the word 'the'*/
        for(i=0;i<=count-3;i++)
        {
                t=(str[i]=='t'||str[i]=='T');
                h=(str[i+1]=='h'||str[i+1]=='H');
                e=(str[i+2]=='e'||str[i+2]=='E');
                space=(str[i+3]==' '||str[i+3]=='\0');
                if ((t&&h&&e&&space)==1)
                  times++;
        }
        printf("Frequency of the word \'the\' is %d\n",times);
        getch();
}
/*-------------------------------------------------------
```

Output

```
Enter a string:
The Teacher's day is the birth day of Dr.S.Radhakrishnan
Frequency of the word 'the' is 2
------------------------------------------------------------*/
```

47. Write a C program to compute the value of X ^ N given X and N as inputs

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

```
 void main()
 {
        long int x,n,xpown;
        long int power(int x, int n);
        printf("Enter the values of X and N\n");
        scanf("%ld %ld",&x,&n);
        xpown = power (x,n);
        printf("X to the power N = %ld\n");
 }
/*Recursive function to computer the X to power N*/
long int power(int x, int n)
{
        if (n==1)
        return(x);
        else if ( n%2 == 0)
        return (pow(power(x,n/2),2));  /*if n is even*/
        else
        return (x*power(x, n-1));     /* if n is odd*/
}
/*-------------------------------------
```

Output

```
Enter the values of X and N
2 5
X to the power N = 32

RUN2
Enter the values offX and N
4 4
X to the power N ==256

RUN3
Enter the values of X and N
5 2
X to the power N = 25

RUN4
Enter the values of X and N
10 5
X to the power N = 100000
----------------------------------------*/
```

48. Write a C program to accept two matrices and find the sum and difference of the matrices

```
#include <stdio.h>
#include <stdlib.h>
int A[10][10], B[10][10], sumat[10][10], diffmat[10][10];
int i, j, R1, C1, R2, C2;
void main()
{

        /* Function declarations */
        void readmatA();
        void printmatA();
        void readmatB();
        void printmatB();
        void sum();
        void diff();
        printf("Enter the order of the matrix A\n");
        scanf("%d %d", &R1, &C1);
        printf("Enter the order of the matrix B\n");
        scanf("%d %d", &R2,&C2);
        if( R1 != R2 && C1 != C2)
        {
                printf("Addition and subtraction are possible\n");
                exit(1);
        }
        else
        {
                printf("Enter the elements of matrix A\n");
                readmatA();
                printf("MATRIX A is\n");
                printmatA();
                printf("Enter the elements of matrix B\n");
                readmatB();
                printf("MATRIX B is\n");
                printmatB();
                sum();
                diff();
        }
} /* main() */
/* Function to read a matrix A */
void readmatA()
{
        for(i=0; i<R1; i++)
```

```
        {
                for(j=0; j<C1; j++)
                {
                        scanf("%d",&A[i][j]);
                }
        }
        return;
}
/* Function to read a matrix B */
void readmatB()
{
        for(i=0; i<R2; i++)
        {
                 for(j=0; j<C2; j++)
                 {
                        scanf("%d",&B[i][j]);
                }
        }
}
/* Function to print a matrix A */
void printmatA()
{
        for(i=0; i<R1; i++)
        {
                for(j=0; j<C1; j++)
                {
                        printf("%3d",A[i][j]);
                }
                printf("\n");
        }
}
/* Function to print a matrix B */
void printmatB()
{
        for(i=0; i<R2; i++)
        {
                for(j=0; j<C2; j++)
                {
                        printf("%3d",B[i][j]);
                }
                printf("\n");
        }
}
/*Function to find the sum of elements of matrix A and Matrix B*/
```

```
void sum()
{
        for(i=0; i<R1; i++)
        {
                for(j=0; j<C2; j++)
                {
                        sumat[i][j] = A[i][j] + B[i][j];
                }
        }
        printf("Sum matrix is\n");
        for(i=0; i<R1; i++)
        {
                for(j=0; j<C2; j++)
                {
                        printf("%3d",sumat[i][j]) ;
                }
                printf("\n");
        }
        return;
}
/*Function to find the difference of elements of matrix A and Matrix B*/
void diff()
{
        for(i=0; i<R1; i++)
        {
                for(j=0; j<C2; j++)
                {
                        diffmat[i][j] = A[i][j] - B[i][j];
                }
        }
        printf("Difference matrix is\n");
        for(i=0; i<R1; i++)
        {
                for(j=0; j<C2; j++)
                {
                        printf("%3d",diffmat[i][j]);
                }
                printf("\n");
        }
        return;
}
```

```
/*-------------------------------------------------
```

Output

```
Enter the order of the matrix A
2 2
Enter the order of the matrix B
2 2
Enter the elements of matrix A
1
2
3
4
MATRIX A is
1  2
3  4
Enter the elements of matrix B
2
4
6
8
MATRIX B is
2  4
6  8
Sum matrix is
3  6
9 12
Difference matrix is
 -1 -2
 -3 -4
-------------------------------------------------------*/
```

49. Write a C Program to accept two matrices and check if they are equal

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main()
{
        int A[10][10], B[10][10];
        int i, j, R1, C1, R2, C2, flag =1;
        printf("Enter the order of the matrix A\n");
        scanf("%d %d", &R1, &C1);
        printf("Enter the order of the matrix B\n");
        scanf("%d %d", &R2,&C2);
```

```
printf("Enter the elements of matrix A\n");
for(i=0; i<R1; i++)
{
        for(j=0; j<C1; j++)
        {
                scanf("%d",&A[i][j]);
        }
}
printf("Enter the elements of matrix B\n");
for(i=0; i<R2; i++)
{
        for(j=0; j<C2; j++)
        {
                scanf("%d",&B[i][j]);
        }
}
printf("MATRIX A is\n");
for(i=0; i<R1; i++)
{
        for(j=0; j<C1; j++)
        {
                printf("%3d",A[i][j]);
        }
        printf("\n");
}
printf("MATRIX B is\n");
for(i=0; i<R2; i++)
{
        for(j=0; j<C2; j++)
        {
                printf("%3d",B[i][j]);
        }
        printf("\n");
}
/* Comparing two matrices for equality */
if(R1 == R2 && C1 == C2)
{
        printf("Matrices can be compared\n");
        for(i=0; i<R1; i++)
        {
                for(j=0; j<C2; j++)
                {
                        if(A[i][j] != B[i][j])
                        {
```

```
                                        flag = 0;
                                        break;
                                }
                        }
                }
        }
        else
        {
                printf(" Cannot be compared\n");
                exit(1);
        }
        if(flag == 1 )
        printf("Two matrices are equal\n");
        else
        printf("But,two matrices are not equal\n");

}

/*-------------------------------------------------------
```

Output

```
Enter the order of the matrix A
2 2
Enter the order of the matrix B
2 2
Enter the elements of matrix A
1 2
3 4
Enter the elements of matrix B
1 2
3 4
MATRIX A is
1  2
3  4
MATRIX B is
1  2
3  4
Matrices can be compared
Two matrices are equal
-------------------------------------------------------*/
```

50. Write a C Program to check if a given matrix is an identity matrix

```
#include <stdio.h>
void main()
{
        int A[10][10];
        int i, j, R, C, flag =1;
        printf("Enter the order of the matrix A\n");
        scanf("%d %d", &R, &C);
        printf("Enter the elements of matrix A\n");
        for(i=0; i<R; i++)
        {
                for(j=0; j<C; j++)
                {
                        scanf("%d",&A[i][j]);
                }
        }
        printf("MATRIX A is\n");
        for(i=0; i<R; i++)
        {
                for(j=0; j<C; j++)
                {
                        printf("%3d",A[i][j]);
                }
                printf("\n");
        }
        /* Check for unit (or identity) matrix */
        for(i=0; i<R; i++)
        {
                for(j=0; j<C; j++)
                {
                        if(A[i][j] != 1 && A[j][i] !=0)
                        {
                                flag = 0;
                                break;
                        }
                }
        }
        if(flag == 1 )
        printf("It is identity matrix\n");
        else
        printf("It is not a identity matrix\n");
}
```

/*----------

Output

Run 1
Enter the order of the matrix A
2 2

Enter the elements of matrix A
2 2
1 2
MATRIX A is
2 2
1 2
It is not a identity matrix

Run 2
Enter the order of the matrix A
2 2
Enter the elements of matrix A
1 0
0 1
MATRIX A is
1 0
0 1
It is identity matrix

----------*/

51. Write a C program to accept a matrix of given order and interchnge any two rows and columns in the original matrix

```
#include <stdio.h>
void main()
{
        static int m1[10][10],m2[10][10];
        int i,j,m,n,a,b,c, p, q, r;
        printf ("Enter the order of the matrix\n");
        scanf ("%d %d",&m,&n);
        printf ("Enter the co-efficents of the matrix\n");
        for (i=0; i<m;++i)
        {
                for (j=0;j<n;++j)
                {
```

```
                scanf ("%d,",&m1[i][j]);
                m2[i][j] = m1[i][j];
        }
}
printf ("Enter the numbers of two rows to be exchnged \n");
scanf ("%d %d", &a,&b);
for (i=0;i<m;++i)
{
        c = m1[a-1][i];   /* first row has index is 0 */
        m1[a-1][i] = m1[b-1][i];
        m1[b-1][i] = c;
}
printf ("Enter the numbers of two columns to be exchnged\n");
scanf ("%d %d",&p,&q);
printf ("The given matrix is \n");
for (i=0;i<m;++i)
{
        for (j=0;j<n;++j)
        printf (" %d",m2[i][j]);
        printf ("\n");
}
for (i=0;i<n;++i)
{
        r = m2[i][p-1]; /* first column index is 0 */
        m2[i][p-1] = m2[i][q-1];
        m2[i][q-1] = r;
}
printf ("The matix after interchnging the two rows(in the original matrix)\n");
for (i=0; i<m; ++i)
{
        for (j=0; j<n; ++j)
        {
                printf (" %d",m1[i][j]);
        }
        printf ("\n");
}
printf("The matix after interchnging the two columns(in the original matrix)\n");
for (i=0;i<m;++i)
{
        for (j=0;j<n;++j)
        {
                printf (" %d", m2[i][j]);
                printf ("\n");
        }
```

```
        }
}

/*---------------------------------------------------------------------
Enter the order of the matrix
3 3
Enter the co-efficents of the matrix
1 2 4
5 7 9
3 0 6
Enter the numbers of two rows to be exchnged
1 2
Enter the numbers of two columns to be exchnged
2 3
The given matrix is
 1 2 4
 5 7 9
 3 0 6
The matix after interchnging the two rows (in the original matrix)
 5 7 9
 1 2 4
 3 0 6
The matix after interchnging the two columns(in the original matrix)
 1 4 2
 5 9 7
 3 6 0
---------------------------------------------------------------------
```

52. Write a C program to display the inventory of items in a store/shop The inventory maintains details such as name, price, quantity and manufacturing date of each item.

```
#include <stdio.h>
#include <conio.h>
void main()
{
        struct date
        {
                int day;
                int month;
                int year;
        };
        struct details
        {
                char name[20];
```

```
        int price;
        int code;
        int qty;
        struct date mfg;
    };
    struct details item[50];
    int n,i;
    clrscr();
    printf("Enter number of items:");
    scanf("%d",&n);
    fflush(stdin);
    for(i=0;i<n;i++)
    {
        fflush(stdin);
        printf("Item name:");
        scanf("%[^\n]",item[i].name);
        fflush(stdin);
        printf("Item code:");
        scanf("%d",&item[i].code);
        fflush(stdin);
        printf("Quantity:");
        scanf("%d",&item[i].qty);
        fflush(stdin);
        printf("price:");
        scanf("%d",&item[i].price);
        fflush(stdin);
        printf("Manufacturing date(dd-mm-yyyy):");
        scanf("%d-%d-%d",&item[i].mfg.day,&item[i].mfg.month,&item[i].mfg.year);
    }
    printf("          *****  INVENTORY *****\n");
    printf("-----------------------------------------------------------\n");
    printf("S.N.|   NAME        |  CODE  | QUANTITY |  PRICE  |MFG.DATE\n");
    printf("-----------------------------------------------------------\n");
    for(i=0;i<n;i++)
    {
        printf("%d  %-15s  %-d  %-5d %-5d %d/%d/%d\n",i+1, item[i].name, item[i].code,
        item[i]. qty,item[i].price,item[i].mfg.day, item[i].mfg.month, item[i].mfg.year);
    }
        printf("--------------------------- ---------------------------------\n");
        getch();
}
```

```
/*-------------------------------------------------
Enter number of items:5
Item name:Tea Powder
Item code:123
Quantity:23
price:40
Manufacturing date(dd-mm-yyyy):12-03-2007
Item name:Milk Powder
Item code:345
Quantity:20
price:80
Manufacturing date(dd-mm-yyyy):30-03-2007
Item name:Soap Powder
Item code:510
Quantity:10
price:30
Manufacturing date(dd-mm-yyyy):01-04-2007
Item name:Washing Soap
Item code:890
Quantity:25
price:12
Manufacturing date(dd-mm-yyyy):10-03-2007
Item name:Shampo
Item code:777
Quantity:8
price:50
Manufacturing date(dd-mm-yyyy):17-05-2007
                                ***** INVENTORY *****
-------------------------------------------------------------------------------
S.N.|   NAME        |  CODE  |  QUANTITY |  PRICE  |MFG.DATE
-------------------------------------------------------------------------------
 1     Tea Powder      123        23            40       12/3/2007
 2     Milk Powder     345        20            80       30/3/2007
 3     Soap Powder     510        10            30       1/4/2007
 4     Washing Soap    890        25            12       10/3/2007
 5     Shampo          777         8            50       17/5/2007
-------------------------------------------------------------------------------          */
```

53. Write a c program to multifly given number by 4 using bitwise operators

```
#include <stdio.h>
void main()
{
        long number, tempnum;
```

```
        printf("Enter an integer\n");
        scanf("%ld",&number);
        tempnum = number;
        number = number << 2;
        /*left shift by two bits*/
        printf("%ld x 4 = %ld\n", tempnum,number);
}

/*------------------------------
Output
Enter an integer
15
15 x 4 = 60

RUN2
Enter an integer
262
262 x 4 = 1048
---------------------------------*/
```

54. Write a C program to accept a set of numbers and compute mean, varience and standard deviation

```
#include <stdio.h>
#include <math.h>
void main()
{
        float x[10];
        int  i, n;
        float avrg, var, SD, sum=0, sum1=0;
        printf("Enter howmany elements\n");
        scanf("%d", &n);
        for(i=0; i<n; i++)
        {
                scanf("%f", &x[i]);
        }
        /* Compute the sum of all elements */
        for(i=0; i<n; i++)
        {
                sum = sum + x[i];
        }
        avrg = sum /(float) n;
        /* Compute  varaience  and standard deviation  */
        for(i=0; i<n; i++)
        {
```

```
                sum1 = sum1 + pow((x[i] - avrg),2);
        }
        var = sum1 / (float) n;
        SD = sqrt(var);
        printf("Avrage of all elements =%.2f\n", avrg);
        printf("Varience of all elements =%.2f\n", avrg);
        printf("Standard deviation of all elements =%.2f\n", SD);

}
/*-------------------------------------------------
Output
Enter howmany elements
5
10
21
32
59
17
Avrage of all elements =27.80
Varience of all elements =27.80
Standard deviation of all elements =17.15
-------------------------------------------------------*/
```

55. Write a C program to accept N numbers and arrange them in an asceding order

```
#include <stdio.h>
void main ()
{
        int i,j,a,n,number[30];
        printf ("Enter the value of N\n");
        scanf ("%d", &n);
        printf ("Enter the numbers \n");
        for (i=0; i<n; ++i)
        scanf ("%d",&number[i]);
        for (i=0; i<n; ++i)
        {
                for (j=i+1; j<n; ++j)
                {
                        if (number[i] > number[j])
                        {
                                a= number[i];
                                number[i] = number[j];
                                number[j] = a;
                        }
                }
```

```
        }
        printf ("The numbers arrenged in ascending order are given below\n");
        for (i=0; i<n; ++i)
        printf ("%d\n",number[i]);
}   /* End of main() */

/*-------------------------------------------------------------------
```

Output

```
Enter the value of N
5
Enter the numbers
80
20
67
10
45

The numbers arrenged in ascending order are given below
10
20
45
67
80
-------------------------------------------------------------------*/
```

56. Write a C program to accept a list of data items and find the second largest and second smallest elements in it. And also computer the average of both. And search for the average value whether it is present in the array or not. Display appropriate message on successful search. */

```
#include <stdio.h>
void main ()
{
        int number[30];
        int i,j,a,n,counter,ave;
        printf ("Enter the value of N\n");
        scanf ("%d", &n);
        printf ("Enter the numbers \n");
        for (i=0; i<n; ++i)
        scanf ("%d",&number[i]);
        for (i=0; i<n; ++i)
        {
                for (j=i+1; j<n; ++j)
                {
```

```
                        if (number[i] < number[j])
                        {
                                a  =  number[i];
                                number[i] = number[j];
                                number[j] = a;
                        }
                }
        }
        printf ("The numbers arranged in descending order are given below\n");
        for (i=0; i<n; ++i)
        {
                printf ("%d\n",number[i]);
        }
        printf ("The 2nd largest number is  = %d\n", number[1]);
        printf ("The 2nd smallest number is = %d\n", number[n-2]);
        ave = (number[1] +number[n-2])/2;
        counter = 0;
        for (i=0; i<n; ++i)
        {
                if (ave == number[i])
                {
                        ++counter;
                }
        }
        if (counter == 0 )
        printf ("The average of %d  and %d is = %d is not in the array\n", number[1], number[n-2],
        ave);
        else
        printf ("The average of %d  and %d in array is %d in numbers\n",number[1], number[n-2],
        counter);
}  /* End of main() */

/*-------------------------------------------------------
Output
Enter the value of N
6
Enter the numbers
30
80
10
40
70
90
```

```
The numbers arranged in descending order are given below
90
80
70
40
30
10
The 2nd largest number is  = 80
The 2nd smallest number is = 30
The average of 80  and 30 is = 55 is not in the array

----------------------------------------------------------*/
```

57. Write a C program to accept an array of integers and delete the specified integer from the list

```
#include <stdio.h>
void main()
{
        int  vectx[10];
        int  i, n, pos, element, found = 0;
        printf("Enter how many elements\n");
        scanf("%d", &n);
        printf("Enter the elements\n");
         for(i=0; i<n; i++)
        {
                scanf("%d", &vectx[i]);
        }
        printf("Input array elements are\n");
        for(i=0; i<n; i++)
        {
                printf("%d\n", vectx[i]);
        }
        printf("Enter the element to be deleted\n");
        scanf("%d",&element);
        for(i=0; i<n; i++)
        {
                if ( vectx[i] == element)
                {
                        found = 1;
                        pos = i;
                        break;
                }
```

```
        }
        if (found == 1)
        {
                for(i=pos; i< n-1; i++)
                {
                        vectx[i] = vectx[i+1];
                }
               printf("The resultant vector is \n");
                for(i=0; i<n-1; i++)
               {
                        printf("%d\n",vectx[i]);
                }
        }
        else
        printf("Element %d is not found in the vector\n", element);

}  /* End of main() */

/*-------------------------------------------------
```

Output

```
Run 1
Enter how many elements
5
Enter the elements
30
10
50
20
40
Input array elements are
30
10
50
20
40
Enter the element to be deleted
35
Element 35 is not found in the vector

Run 2
Enter how many elements
4
```

```
Enter the elements
23
10
55
81

Input array elements are
23
10
55
81
Enter the element to be deleted
55
The resultant vector is
23
10
81
-----------------------------------------------------------*/
```

58. Write a C programme (1-D Array) store some elements in it.Accept key & split from that point. Add the first half to the end of second half

```
#include <stdio.h>
void main ()
{
        int number[30];
        int i,n,a,j;
        printf ("Enter the value of n\n");
        scanf ("%d",&n);
        printf ("enter the numbers\n");
        for (i=0; i<n; ++i)
        scanf ("%d", &number[i]);
        printf ("Enter the position of the element to split the array \n");
        scanf ("%d",&a);
        for (i=0; i<a; ++i)
        {
                number[n] = number[0];
                for (j=0; j<n; ++j)
                {
                        number[j] = number[j+1];
                }
        }
```

```
        printf("The resultant array is\n");
        for (i=0; i<n; ++i)
        {
                printf ("%d\n",number[i]);
        }
}       /* End of main() */

/*-------------------------------------------------
```

Output

```
Enter the value of n
5
enter the numbers
30
10
40
50
60
Enter the position of the element to split the array
2
The resultant array is
40
50
60
30
10
-------------------------------------------------------------*/
```

59. Write a C program to accept a figure code and find the ares of different geometrical figures such as circle, square, rectangle etc using switch

```
#include <stdio.h>
void main()
{
        int    fig_code;
        float  side,base,length,bredth,height,area,radius;
        printf("-------------------------\n");
        printf(" 1 --> Circle\n");
        printf(" 2 --> Rectangle\n");
        printf(" 3 --> Triangle\n");
        printf(" 4 --> Square\n");
        printf("-------------------------\n");
        printf("Enter the Figure code\n");
        scanf("%d",&fig_code);
```

```
        switch(fig_code)
        {
                case 1:  printf("Enter the radius\n");
                         scanf("%f",&radius);
                         area=3.142*radius*radius;
                         printf("Area of a circle=%f\n", area);
                         break;
                case 2:  printf("Enter the bredth and length\n");
                         scanf("%f %f",&bredth, &length);
                         area=bredth *length;
                         printf("Area of a Reactangle=%f\n", area);
                         break;
                case 3:  printf("Enter the base and height\n");
                         scanf("%f %f",&base,&height);
                         area=0.5 *base*height;
                         printf("Area of a Triangle=%f\n", area);
                         break;
                case 4:  printf("Enter the side\n");
                         scanf("%f",&side);
                         area=side * side;
                         printf("Area of a Square=%f\n", area);
                         break;
                default: printf("Error in figure code\n");
                         break;
        }   /* End of switch */
}   /* End of main() */

/*--------------------------------------------------
Output
Run 1
------------------------
 1 --> Circle
 2 --> Rectangle
 3 --> Triangle
 4 --> Square
------------------------
Enter the Figure code
2
Enter the bredth and length
2
6
Area of a Reactangle=12.000000
```

```
Run 2
------------------------
 1 --> Circle
 2 --> Rectangle.
 3 --> Triangle
 4 --> Square
------------------------
Enter the Figure code
3
 Enter the base and height
5
7
Area of a Triangle=17.500000
-----------------------------------------------------*/
```

60.Write a C program to accept a string and find the sum of all digits present in the string

```
#include <stdio.h>
void main()
{
        char string[80];
        int count,nc=0,sum=0;
        printf("Enter the string containing both digits and alphabet\n");
        scanf("%s",string);
        for(count=0;string[count]!='\0'; count++)
        {
                if((string[count]>='0') && (string[count]<='9'))
                {
                        nc += 1;
                        sum += (string[count] - '0');
                }       /* End of if */
        }       /* End of For */
        printf("NO. of Digits in the string=%d\n",nc);
        printf("Sum of all digits=%d\n",sum);
}       /* End of main() */

/*-----------------------------------------------------
```

Output

```
Enter the string containing both digits and alphabet
goodmorning25
NO. of Digits in the string=2
Sum of all digits=7

-----------------------------------------------------*/
```

61. Write a C program to accept a matric of order MxN and find its transpose

```
#include <stdio.h>
void main ()
{
        static int ma[10][10];
        int i,j,m,n;
        printf ("Enter the order of the matrix \n");
        scanf ("%d %d",&m,&n);
        printf ("Enter the coefiicients of the matrix\n");
        for (i=0;i<m;++i)
        {
                for (j=0;j<n;++j)
                {
                        scanf ("%d",&ma[i][j]);
                }
        }
        printf ("The given matrix is \n");
        for (i=0;i<m;++i)
        {
                for (j=0;j<n;++j)
                {
                        printf (" %d",ma[i][j]);
                }
                printf ("\n");
        }
        printf ("Transpose of matrix is \n");
        for (j=0;j<n;++j)
        {
                for (i=0;i<m;++i)
                {
                        printf (" %d",ma[i][j]);
                }
                printf ("\n");
        }
}       /* End of main() */

/*----------------------------------------
```

Output

```
Enter the order of the matrix
2 2
Enter the coefiicients of the matrix
3 -1
6 0
```

```
The given matrix is
 3 -1
 6 0
Transpose of matrix is
 3 6
 -1 0
-----------------------------------------*/
```

62. Write a C program to accept a amtric of order MxN and sort all rows of the matrix in ascending order and all columns in descendng order /

```
#include <stdio.h>
void main ()
{
        static int ma[10][10],mb[10][10];
        int i,j,k,a,m,n;
        printf ("Enter the order of the matrix \n");
        scanf ("%d %d", &m,&n);
        printf ("Enter co-efficients of the matrix \n");
        for (i=0;i<m;++i)
        {
                for (j=0;j<n;++j)
                {
                        scanf ("%d",&ma[i][j]);
                        mb[i][j] = ma[i][j];
                }
        }
        printf ("The given matrix is \n");
        for (i=0;i<m;++i)
        {
                for (j=0;j<n;++j)
                {
                        printf (" %d",ma[i][j]);
                }
                printf ("\n");
        }
        printf ("After arranging rows in ascending order\n");
        for (i=0;i<m;++i)
        {
                for (j=0;j<n;++j)
                {
                        for (k=(j+1);k<n;++k)
                        {
                                if (ma[i][j] > ma[i][k])
                                {
                                        a = ma[i][j];
                                        ma[i][j] = ma[i][k];
                                        ma[i][k] = a;
```

```
                        }
                    }
                }
        }   /* End of outer for loop*/
        for (i=0;i<m;++i)
        {
                for (j=0;j<n;++j)
                {
                        printf (" %d",ma[i][j]);
                }
                printf ("\n");
        }
        printf ("After arranging the columns in descending order \n");
        for (j=0;j<n;++j)
        {
                for (i=0;i<m;++i)
                {
                        for (k=i+1;k<m;++k)
                        {
                                if (mb[i][j] < mb[k][j])
                                {
                                        a = mb[i][j];
                                        mb[i][j] = mb[k][j];
                                        mb[k][j] = a;
                                }
                        }
                }
        }   /* End of outer for loop*/
        for (i=0;i<m;++i)
        {
                for (j=0;j<n;++j)
                {
                        printf (" %d",mb[i][j]);
                }
                printf ("\n");
        }
} /*End of main() */

/*-------------------------------------------------
```

Output

```
Enter the order of the matrix
2 2
Enter co-efficients of the matrix
3 1
```

```
5 2
The given matrix is
3 1
5 2
After arranging rows in ascending order
1 3
2 5
After arranging the columns in descending order
52
31
```

63. Write a C program to accept a set of names and sort them in an alphabetical order, Use structures to store the names

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct person
{
        char name[10];
        int rno;
};
typedef struct person NAME;
NAME  stud[10], temp[10];
void main()
{
        int no,i;
        void sort(int N);
        /* Function declaration */
        clrscr();
        fflush(stdin);
        printf("Enter the number of students in the list\n");
        scanf("%d",&no);
        for(i = 0; i < no; i++)
        {
                printf("\nEnter the name of  person %d : ", i+1);
                fflush(stdin);
                gets(stud[i].name);
                printf("Enter the roll number of %d : ", i+1);
                scanf("%d",&stud[i].rno);
                temp[i] = stud[i];
        }
        printf("\n**************************\n");
        printf ("     Names before sorting     \n");
```

```
        /* Print the list of names before sorting */
        for(i=0;i<no;i++)
        {
                printf("%-10s\t%3d\n",temp[i].name,temp[i].rno);
        }
        sort(no);       /* Function call */
        printf("\n*****************************\n");
        printf ("     Names after sorting     \n");
        printf("\n*****************************\n");
        /* Display the sorted names */
        for(i=0;i<no;i++)
        {
                printf("%-10s\t%3d\n",stud[i].name,stud[i].rno);
        }
        printf("\n*****************************\n");
}       /* End of main() */
/* Function to sort the given names */
void sort(int N)
{
        int i,j;
        NAME temp;
        for(i = 0; i < N-1;i++)
        {
                for(j = i+1; j < N; j++)
                {
                        if(strcmp(stud[i].name,stud[j].name) > 0 )
                        {
                                temp    = stud[i];
                                stud[i] = stud[j];
                                stud[j] = temp;
                        }
                }
        }
}   /* end of sort() */

/*-------------------------------------------------------
```

Output

```
Enter the number of students in the list
5
Enter the name of  person 1 : Rajashree
Enter the roll number of 1 : 123

Enter the name of  person 2 : John
Enter the roll number of 2 : 222
```

```
Enter the name of  person 3 : Priya
Enter the roll number of 3 : 331

Enter the name of  person 4 : Anand
Enter the roll number of 4 : 411

Enter the name of  person 5 : Nirmala
Enter the roll number of 5 : 311

*****************************
  Names before sorting
Rajashree       123
John            222
Priya           331
Anand           411
Nirmala         311

*****************************
    Names after sorting

*****************************
Anand           411
John            222
Nirmala         311
Priya           331
Rajashree       123

*****************************
--------------------------------------------------*/
```

64. Write a C Program to convert the lower case letters to upper case and vice-versa

```
#include <stdio.h>
#include <ctype.h>
#include <conio.h>
void main()
{
        char sentence[100];
        int count, ch, i;
        clrscr();
        printf("Enter a sentence\n");
        for(i=0; (sentence[i] = getchar())!='\n'; i++)
        {
```

```
                ;
        }
        count = i;
        printf("\nInput sentence is     : %s ",sentence);
        printf("\nResultant sentence is : ");
        for(i=0; i < count; i++)
        {
                ch = islower(sentence[i]) ? toupper(sentence[i]) :  tolower(sentence[i]);putchar(ch);
        }

} /*End of main()

/*----------------------------------------
```

Output

```
Enter a sentence
gOOD mORNING

Input sentence is       : gOOD mORNING

Resultant sentence is : Good Morning
----------------------------------------*/
```

65.Write a C program to find the sum of two one-dimensional arrays using Dynamic Memory Allocation

```
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
void main()
{
        int i,n;
        int *a,*b,*c;
        printf("How many Elements in each array...\n");
        scanf("%d", &n);
        a = (int *) malloc(n*sizeof(int));
        b = (int *) malloc(n*sizeof(int));
        c =( int *) malloc(n*sizeof(int));
        printf("Enter Elements of First List\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",a+i);
        }
        printf("Enter Elements of Second List\n");
        for(i=0;i<n;i++)
        {
```

```
                scanf("%d",b+i);
        }
        for(i=0;i<n;i++)
        {
                *(c+i) = *(a+i) + *(b+i);
        }
        printf("Resultant List is\n");
        for(i=0;i<n;i++)
        {
                printf("%d\n",*(c+i));
        }

}     /* End of main() */

/*---------------------------------------
```

Output

```
How many Elements in each array...
4
Enter Elements of First List
1
2
3
4
Enter Elements of Second List
6
7
8
9
Resultant List is
7
9
11
13

---------------------------------------*/
```

66. Write a C program to illustrate the concept of unions

```
#include <stdio.h>
#include <conio.h>
void main()
{
        union number
        {
                int  n1;
```

```
            float n2;
        };
        union number x;
        clrscr() ;
        printf("Enter the value of n1: ");
        scanf("%d", &x.n1);
        printf("Value of n1 =%d", x.n1);
        printf("\nEnter the value of n2: ");
        scanf("%d", &x.n2);
        printf("Value of n2 = %d\n",x.n2);

}       /* End of main() */

/*-----------------------------------
Output
Enter the value of n1: 2
Value of n1 =2
Enter the value of n2: 3
Value of n2 = 0
-----------------------------------*/
```

67.Write a C program to create a linked list and display the elements in the list.

```
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
void main()
{
        struct node
        {
                int num;
                struct node *ptr;
        };
        typedef struct node NODE;
        NODE *head, *first, *temp=0;
        int count = 0;
        int choice = 1;
        first = 0;
        while(choice)
        {
                head =(NODE*) malloc(sizeof(NODE));
                printf("Enter the data item\n");
                scanf("%d", &head-> num);
                if(first != 0)
```

```
                {
                        temp->ptr = head;
                        temp = head;
                }
                else
                {
                        first = temp = head;
                }
                fflush(stdin);
                printf("Do you want to continue(Type 0 or 1)?\n");
                scanf("%d", &choice);
        }       /* End of while */
        temp->ptr = 0;
        temp = first;    /* reset temp to the beginning*/
        printf("\nstatus of the linked list is\n");
        while(temp!=0)
        {
                printf("%d=>", temp->num);
                count++;
                temp = temp -> ptr;
        }
        printf("NULL\n");
        printf("No. of nodes in the list = %d\n", count);
}               /* End of main*/

/*-----------------------------------------------
```

Output

```
Enter the data item
10
Do you want to continue(Type 0 or 1)?
1
Enter the data item
34
Do you want to continue(Type 0 or 1)?
1
Enter the data item
56
Do you want to continue(Type 0 or 1)?
0

status of the linked list is
10=>34=>56=>NULL
No. of nodes in the list = 3
-----------------------------------------------*/
```

68. Write a C program to illustrate the operations of singly linked list

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#define MAX 30
struct EMP
{
        int  empno;
        char empName[MAX];
        char designation[MAX];
        struct EMP *next;
};
************************************************/
/* Function to insert a node at the front of the linked list. */
/* front: front pointer, id: employee ID, name: employee name */
/* desg: Employee designation          */
/* Returns the new front pointer.      */
/***********************************************
struct EMP* insert(struct EMP *front, int id, char name[], char desg[])
{
        struct EMP *newnode;
        newnode = (struct EMP*) malloc(sizeof(struct EMP));
        if (newnode == NULL)
        {
                printf("\nAllocation failed\n");
                exit(2);
        }
        newnode->empno = id;
        strcpy(newnode->empName, name);
        strcpy(newnode->designation, desg);
        newnode->next = front;
        front = newnode;
        return(front);
}   /*End of insert() */
/* Function to display a node in a linked list */
void printNode(struct EMP *p)
{
        printf("\nEmployee Details...\n");
        printf("\nEmp No         : %d", p->empno);
        printf("\nName          : %s", p->empName);
        printf("\nDesignation    : %s\n", p->designation);
        printf("-----------------------------------\n");
```

```
}                   /*End of printNode() */
/* Function to deleteNode a node based on employee number     */
/* front: front pointer, id: Key value                  */
/* Returns: the modified list.                          */
/* ************************************************
struct EMP* deleteNode(struct EMP *front, int id)
{
        struct EMP *ptr;
        struct EMP *bptr;    /* bptr is pointing to the node behind ptr */
        if (front->empno == id)
        {
                ptr = front;
                printf("\nNode deleted:");
                printNode(front);
                front = front->next;
                free(ptr);
                return(front);
        }
        for(ptr=front->next, bptr=front; ptr!=NULL; ptr=ptr->next, bptr=bptr->next)
        {
                if (ptr->empno == id)
                {
                        printf("\nNode deleted:");
                        printNode(ptr);
                        bptr->next = ptr->next;
                        free(ptr);
                        return(front);
                }
        }
        printf("\nEmployee Number %d not found ", id);
        return(front);
}       /*End of deleteNode() */
/************************************************/* Function to search the nodes in a
linear fashion based emp ID */
/* front: front pointer, key: key ID.                   */
/************************************************void search(struct EMP *front, int key)
{
        struct EMP *ptr;
        for (ptr = front; ptr != NULL; ptr = ptr -> next)
        {
                if (ptr->empno == key)
                {
                        printf("\nKey found:");
```

```
                        printNode(ptr);
                        return;
                }
        }
        printf("\nEmployee Number %d not found ", key);
}    /*End of search() */
/* Function to display the linked list */
void display(struct EMP *front)
{
        struct EMP *ptr;
        for (ptr = front; ptr != NULL; ptr = ptr->next)
        {
                printNode(ptr);
        }
}       /*End of display() */
/* Function to display the menu of operations on a linked list */
void menu()
{
        printf("-----------------------------------------------\n");
        printf("Press 1 to INSERT a node into the list      \n");
        printf("Press 2 to DELETE a node from the list      \n");
        printf("Press 3 to DISPLAY the list                 \n");
        printf("Press 4 to SEARCH the list                  \n");
        printf("Press 5 to EXIT                           \n");
        printf("-----------------------------------------------\n");
}               /*End of menu() */
/* Function to select the option */
char option()
{
        char choice;
        printf("\n\n>> Enter your choice: ");
        switch(choice=getche())
        {
                case '1' :
                case '2' :
                case '3' :
                case '4' :
                case '5' :   return(choice);
                default :   printf("\nInvalid choice.");
         }
         return choice;
} /*End of option() */
/* The main() program begins */
void main()
```

```
{
        struct EMP *linkList;
        char name[21], desig[51];
        char choice;
        int eno;
        linkList = NULL;
        printf("\nWelcome to demonstration of singly linked list\n");
        menu();                 /*Function call */
        do
        {
                choice = option();
                /*to choose oeration to be performed */
                switch(choice)
                {
                        case '1':printf("\nEnter the Employee Number      : ");
                                scanf("%d", &eno);
                                printf("Enter the Employee name       : ");
                                fflush(stdin);
                                gets(name);
                                printf("Enter the Employee Designation : ");
                                gets(desig);
                                linkList = insert(linkList, eno, name, desig);
                                break;
                        case '2':printf("\n\nEnter the employee number to be deleted: ");
                                scanf("%d", &eno);
                                linkList = deleteNode(linkList, eno);
                                break;
                        case '3':if (linkList == NULL)
                                printf("\nList empty.");
                                break;
                                display(linkList);
                                break;
                        case '4':printf("\n\nEnter the employee number to be searched: ");
                                scanf("%d", &eno);
                                search(linkList, eno);
                                break;
                        case '5':break;
                }
        } while (choice != '5');
}/*End fo main()*/
```

/*---

Output

```
Welcome to demonstration of singly linked list
----------------------------------------------
Press 1 to INSERT a node into the list
Press 2 to DELETE a node from the list
Press 3 to DISPLAY the list
Press 4 to SEARCH the list
Press 5 to EXIT
----------------------------------------------
>> Enter your choice: 1
Enter the Employee Number          : 1234
Enter the Employee name            : Keerthi
Enter the Employee Designation     : Engineer

>> Enter your choice: 1
Enter the Employee Number          : 2345
Enter the Employee name            : Srinivasan
Enter the Employee Designation     : Specilist

>> Enter your choice: 1
Enter the Employee Number          : 4567
Enter the Employee name            : Annapoorna
Enter the Employee Designation     : Project Manager

>> Enter your choice: 3
Employee Details...

Emp No       : 4567
Name         : Annapoorna
Designation  : Project Manager
-------------------------------------

Employee Details...

Emp No       : 2345
Name         : Srinivasan
Designation  : Specilist
-------------------------------------

Employee Details...

Emp No       : 1234
```

```
Name        : Keerthi
Designation : Engineer
-------------------------------------
>> Enter your choice: 2
Enter the employee number to be deleted: 2345

Node deleted:
Employee Details...

Emp No      : 2345
Name        : Srinivasan
Designation : Specilist
-------------------------------------


>> Enter your choice: 3
Employee Details...

Emp No      : 4567
Name        : Annapoorna
Designation : Project Manager
-------------------------------------

Employee Details...

Emp No      : 1234
Name        : Keerthi
Designation : Engineer
-------------------------------------


>> Enter your choice: 4
Enter the employee number to be searched: 2345
Employee Number 2345 not found

>> Enter your choice: 5

------------------------------------------------------------------------*/
```

69. Write a C program to implement stack. Stack is a LIFO data strcuture

LIFO - Last in First Out
Perform PUSH(insert operation), POP(Delete operation) and Display stack

```
#include <stdio.h>
#include <conio.h>
#define MAXSIZE 5
struct stack  /* Structure definition for stack */
{
        int stk[MAXSIZE];
        int top;
};
typedef struct stack STACK;
STACK s;
/* Function declaration/Prototype*/
void push (void);
int  pop(void);
void display (void);
void main ()
{
        int choice;
        int option = 1;
        clrscr ();
        s.top = -1;
        printf ("STACK OPERATION\n");
        while (option)
        {
                printf ("-----------------------------------------\n");
                printf ("       1       -->     PUSH          \n");
                printf ("       2       -->     POP           \n");
                printf ("       3       -->     DISPLAY       \n");
                printf ("       4       -->     EXIT          \n");
                printf ("-----------------------------------------\n");
                printf ("Enter your choice\n");
                scanf   ("%d", &choice);
                switch (choice)
                {
                        case 1: push();
                                break;
                        case 2: pop();
                                break;
                        case 3: display();
                                break;
                        case 4: return;
                }
                fflush (stdin);
                printf ("Do you want to continue(Type 0 or 1)?\n");
                scanf   ("%d", &option);
```

```
        }
}
/*Function to add an element to the stack*/
void push ()
{
        int num;
        if (s.top == (MAXSIZE - 1))
        {
                printf ("Stack is Full\n");
                return;
        }
        else
        {
                printf ("Enter the element to be pushed\n");
                scanf ("%d", &num);
                s.top = s.top + 1;
                s.stk[s.top] = num;
        }
        return;
}
/*Function to delete an element from the stack*/
int pop ()
{
        int num;
        if (s.top == - 1)
        {
                printf ("Stack is Empty\n");
                return (s.top);
        }
        else
        {
                num = s.stk[s.top];
                printf ("poped element is = %d\n", s.stk[s.top]);
                s.top = s.top - 1;
        }
        return(num);
}
/*Function to display the status of the stack*/
void display ()
{
        int i;
        if (s.top == -1)
        {
                printf ("Stack is empty\n");
```

```
                return;
        }
        else
        {
                printf ("\nThe status of the stack is\n");
                for (i = s.top; i >= 0; i--)
                {
                        printf ("%d\n", s.stk[i]);
                }
        }
        printf ("\n");
}

/*-------------------------------------------------------------------------
```

Output

```
STACK OPERATION
----------------------------------------
1       -->     PUSH
2       -->     POP
3       -->     DISPLAY
 4       -->     EXIT
----------------------------------------
Enter your choice
1
Enter the element to be pushed
23
Do you want to continue(Type 0 or 1)?
1
----------------------------------------
1       -->     PUSH
2       -->     POP
3       -->     DISPLAY
4       -->     EXIT
----------------------------------------
Enter your choice
1
Enter the element to be pushed
45
Do you want to continue(Type 0 or 1)?
1
----------------------------------------
1       -->     PUSH
2       -->     POP
3       -->     DISPLAY
```

```
4       -->     EXIT
-----------------------------------------

Enter your choice
1
Enter the element to be pushed
78
Do you want to continue(Type 0 or 1)?
1
-----------------------------------------
1       -->     PUSH
2       -->     POP
3       -->     DISPLAY
4       -->     EXIT
-----------------------------------------
Enter your choice
3
The status of the stack is
78
45
23
Do you want to continue(Type 0 or 1)?
1
-----------------------------------------
1       -->     PUSH
2       -->     POP
3       -->     DISPLAY
4       -->     EXIT
-----------------------------------------
Enter your choice
2
poped element is = 78
Do you want to continue(Type 0 or 1)?
1
-----------------------------------------
1       -->     PUSH
2       -->     POP
3       -->     DISPLAY
4       -->     EXIT
-----------------------------------------
Enter your choice
3
```

```
The status of the stack is
45
23

Do you want to continue(Type 0 or 1)?
0
---------------------------------------- * /
```

APPENDI X – G

DATA STRUCTURE PROGRAMS

1. STACK USING ARRAY

```
#include<stdio.h>
#include<conio.h>
struct stack
{
        int a[10];
        int top;
};
struct stack s1;
void main()
{
        int i,ch,elem=0,j;
        clrscr();
        s1.top=-1;
        do
        {
                clrscr();
                printf("\n choice");
                printf("\n 1.push \n 2.pop \n 3.exit:\n");
                printf("\n Enter your choice:");
                scanf("\n \n %d",&ch);
                switch(ch)
                {
                case 1:if(s1.top<9)
                        {
                                s1.top++;
                                printf("\n Enter the no to push op:");
                                scanf("\n %d",&elem);
                                s1.a[s1.top]=elem;
                                for(i=s1.top;i>=0;i--)
                                printf("\n %d",s1.a[i]);
                        }
                        else
                        printf("\n can't push");
                        break;
                case 2:if(s1.top==-1)
                        {
                                printf("\n stack is empty");
                                break;
                        }
                        else
                        {
                                s1.top--;
                                for(i=s1.top;i>=0;i--)
                                printf("\n %d",s1.a[i]);
                        }
```

```
                    break;
                case 3: exit(0);
                    break;
                default:printf("This is invalid choice");
                     break;
                }
                printf("\n \n Do u want to continue(y/n)?1/2:");
                scanf("%d",&j);
        }while(j!=2);
        getch();
}
```

2. STACK USING LINK LIST

```
#include<stdio.h>
#include<conio.h>
struct link
{
        int info;
        struct link *next;
};
struct link *t,*start,*temp;
void main()
{
        int ele,ch,i,pos,pos1;
        start=NULL;
        while(1)
        {
                clrscr();
                printf("\n\t\t 1.Insertion  \n\t\t 2.Deletion front");
                printf("\n\t\t 3.display \n\t\t 4.Exit");
                printf("\n\n Enter your choice: ");
                scanf("%d",&ch);
                switch(ch)
                {
                case 1:
                        /*Insert a node at front*/
                        t=(struct link *) malloc(sizeof(struct link));
                        printf("\n enter the element: ");
                        scanf("%d",&ele);
                        t->info=ele;
                        t->next=start;
                        start=t;
                        t=start;
                        clrscr();
                        printf("The stack elements");
                        while(t!=NULL)
                        {
                                printf(" ");
```

```
                        printf("\t%d",t->info);
                        t=t->next;
                }
                getch();
                break;
        case 2: /*Deletion a node at front*/
                clrscr();
                t=start;
                start=start->next;
                if(t==NULL)
                {
                        printf("\n\n\t\t Stack is Empty");
                        getch();
                        break;
                }
                clrscr();
                printf("\n\n\t\t One Node %d removed",t->info);
                getch();
                free(t);
                break;
        case 3: /* Display*/
                clrscr();
                t=start;
                while(t!=NULL)
                {
                        printf(" ");
                        printf("\t %d",t->info);
                        t=t->next;
                }
                getch();
                break;
        case 4: exit(0);
                break;
        }
    }
}
```

3.QUEUE USING ARRAY

```
#include<stdio.h>
#include<conio.h>
#define queuesize 5
struct queue
{
        int ele[6];
        int front,rear;
}m;
void main()
{
        int ch,j,i;
        clrscr();
```

```
m.front=m.rear=-1;
do
{
        clrscr();
        printf("\n1.Add \n2.Remove \n3.Display");
        printf("\n\n Enter your choice");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1: if(m.rear==queuesize)
                m.rear=0;
                else
                m.rear++;
                if(m.rear==m.front)
                {
                        printf("\n\t\tQueue is now full");
                        m.rear--;
                        break;
                }
                else
                {
                        printf("\n\t Enter the element");
                        scanf("%d",&m.ele[m.rear]);
                        printf("\n\t\t element is added");
                        getch();
                }
                break;
        case 2: if(m.front!=m.rear)
                {
                        m.front++;
                        if(m.front==queuesize)
                        m.front=0;
                        printf("\n\t\tElement deleted");
                }
                else
                printf("Queue empty");
                break;
        case 3: j=m.front;
                while(j<m.rear)
                {
                        j++;
                        if(j==queuesize)
                             j=0;
                        printf("\n %d",m.ele[j]);
                }
                getch();
                break;
        }
        printf("\n\n Do you want to continue(y/n)?");
        scanf("%d",&i);
```

```
        }while(i!=2);
        getch();
}
```

4. QUEUE USING LINK LIST

```
#include<stdio.h>
#include<conio.h>
struct link
{
        int info;
        struct link *next;
};
struct link *m,*t,*start,*temp;
void main()
{
        int ele,ch,i,pos,pos1;
        start=NULL;
        while(1)
        {
                clrscr();
                printf("\n\t\t 1.Insertion  \n\t\t 2.Deletion ");
                printf("\n\t\t 3.display \n\t\t 4.Exit");
                printf("\n\n Enter your choice: ");
                scanf("%d",&ch);
                switch(ch)
                {
                case 1: /*Insert a node at front*/
                        if(start=='\0')
                        {
                                t=(struct link *)malloc(sizeof(struct link));
                                printf("\n enter the element: ");
                                scanf("%d",&ele);
                                t->info=ele;
                                t->next=start;
                                start=t;
                        }
                        else
                        {
                                temp=start;
                                pos=1;
                                while(temp->next!='\0')
                                {
                                        pos=pos+1;
                                        temp=temp->next;
                                }
                                m=start;
                                for(i=1;i<pos;i++)
                                {
                                        m=m->next;
                                }
```

```
                        t=(struct link *) malloc(sizeof(struct link));
                        printf("\n\n enter the element: ");
                        scanf("\n\n %d",&ele);
                        t->info=ele;
                        t->next=temp->next;
                        temp->next=t;
                }
                break;
        case 2: /*Deletion a node at front*/
                t=start;
                start=start->next;
                if(t==NULL)
                {
                        printf("\n\n\t\t Can't remove");
                        getch();
                        break;
                }
                printf("\n\n\t\t One Node %d removed",t->info);
                getch();
                free(t);
                break;
        case 3: /* Display*/
                clrscr();
                t=start;
                while(t!=NULL)
                {
                        printf(" ");
                        printf("\t %d",t->info);
                        t=t->next;
                }
                getch();
                break;
        case 4: exit(0);
                break;
        }
    }
}
```

5. Complete Link List

```
*Program for Inserting and Deleting element at front&middle using linked
                        list. date 18.10.2005*/

#include<stdio.h>
#include<conio.h>
struct link
{
        int info;
        struct link *next;
};
```

```
struct link *t,*start,*temp;
void main()
{
	int ele,ch,i,pos,pos1,flag;
	start=NULL;
	while(1)
	{
		clrscr();
		printf("\n\n\t\t 1.Insertion front \n\n\t\t 2.Deletion front");
		printf("\n\n\t\t 3.Insertion after \n\n\t\t 4.Deletion after");
		printf("\n\n\t\t 5.searching");
		printf("\n\n\t\t 6.display \n\n\t\t 7.Exit");
		printf("\n\n Enter your choice: ");
		scanf("%d",&ch);
		switch(ch)
		{
		case 1: /*Insert a node at front*/
			t=(struct link *)malloc (sizeof(struct link));
			printf("\n enter the element: ");
			scanf("%d",&ele);
			t->info=ele;
			t->next=start;
			start=t;
			t=start;
			while(t!=NULL)
			{
				printf(" ");
				printf("\t %d",t->info);
				t=t->next;

			}
			getch();
			break;
		case 2: /*Deletion a node at front*/
			t=start;
			start=start->next;
			if(t==NULL)
			{
				printf("\n\n\t\t Can't remove");
				getch();
				break;
			}
			printf("\n\n\t\t One Node %d removed",t->info);
			getch();
			free(t);
			break;
		case 3: /*Insertion a node at middle*/
			printf("\n\n Enter the position: ");
			scanf("%d",&pos);
```

```
            temp=start;
            for(i=1;i<pos;i++)
            {
                    temp=temp->next;
            }
             t=(struct link *)malloc(sizeof(struct link));
            printf("\n\n enter the element: ");
            scanf("\n\n %d",&ele);
            t->info=ele;
            t->next=temp->next;
            temp->next=t;
            t=start;
            while(t!=NULL)
            {
                    printf(" ");
                    printf("\t %d",t->info);
                    t=t->next;
            }
            getch();
            break;
    case 4: /* Deletion a node at middle*/
            printf("\n\n Enter the position: ");
            scanf("%d",&pos1);
            temp=start;
            for(i=1;i<pos1;i++)
            {
                    temp=temp->next;
            }
            t=temp->next;
            temp->next=t->next;
            free(t);
            break;
     case 5:clrscr();
            printf("\n\nSearching an element");
            printf("\n_____");
            printf("\n\nPlease enter the element :");
            scanf("%d",&ele);
            flag=0;
            temp=start;
            while(temp!=NULL)
            {
                    if(temp->info==ele)
                    {
                            printf("\n\nThe element is present");
                            flag++;
                    }//end if
                    temp=temp->next;
            }//end while
            if(flag==0)
            printf("\n\nThe element is not present");
```

```
                getch();
                break;
        case 6: /* Display*/
                clrscr();
                t=start;
                while(t!=NULL)
                {
                        printf(" ");
                        printf("\t %d",t->info);
                        t=t->next;
                }
                getch();
                break;
        case 7: exit(0);
                break;
        }
    }
}
```

6. Circular Link List

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
struct cirlink
{
    int info;
    struct cirlink *next;
};
void main()
{
    struct cirlink *last,*t,*p,*temp;
    int pos,pos1;
    int i,elem,ch,c,ele,flag;
    last=NULL;
    clrscr();
    do
    {
        printf("1.Insert at the front \n\n");
        printf("2.Delete at front\n\n");
        printf("3.Insert After \n\n");
        printf("4.Delete from between \n\n");
        printf("5.Searching an Element in the list\n\n");
        printf("Please enter your choice :");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1: clrscr();
                printf("  \n\t\t\t Insertion at front");
                printf("  \n\t\t\t **************");
                printf("\n\nPlease enter an element  : ");
```

```
        scanf("%d",&elem);
        t=(struct cirlink *) malloc(sizeof(struct cirlink));
        t->info=elem;
        if (last==NULL)
        {
                t->next=t;
                last=t;
        }
        else
        {
                t->next= last->next;
                last->next=t;
        }
        printf("The element in the list is :\n ");
        t=last->next;
        while(t!=last)
        {
                printf("%d\t",t->info);
                t=t->next;
        }
        printf("%d\t",t->info);
        getch();
        break;
case 2: clrscr();
        printf("  \n\t\t\t Deletion at front");
        printf("  \n\t\t\t ******************");
        if(last->next==last)
        last=NULL;
        if(last==NULL)
        printf("\nList is Empty");
        else
        {
                t=last;
                t=last->next;
                last->next=t->next;
                free(t);
                printf("\n DELETED ");
                printf("\nThe element in the list is : \n");
                t=last->next;
                while(t!=last)
                {
                        printf("%d\t",t->info);
                        t=t->next;
                }
                printf("%d\t",t->info);
        }
        getch();
        break;
case 3: clrscr();
        t=(struct cirlink *) malloc(sizeof(struct cirlink));
```

```
        printf("  \n\t\t\t INSERTION AT LAST ");
        printf("  \n\t\t\t ********** \n");
        printf(" At which position you want to insert :");
        scanf("%d",&pos);
        printf("\n\nPlease enter an element  : ");
        scanf("%d",&elem);
        temp=last->next;
        for(i=1;i<pos;i++)
        {
                temp=temp->next;
        }
        t->info=elem;
        t->info=elem;
        t->next=temp->next;
        temp->next=t;
        temp=last->next;
        printf("The element in the list is :\n ");
        while(temp!=last)
        {
                printf("%d\t",temp->info);
                temp=temp->next;
        }
        printf("%d\t",temp->info);
        getch();
        break;
case 4: clrscr();
        printf("  \n\t\t\t DELETION AT LAST ");
        printf("  \n\t\t\t **********\n ");
        if(last->next==last)
        last=NULL;
        if (last->next==NULL)
        {
                printf("List is empty");
        }
        else
        {
                printf("Enter the position you want to Delete :");
                scanf("%d",&pos1);
                temp=last->next;
                for(i=1;i<pos1;i++)
                {
                        temp=temp->next;
                }
                if(temp->next==last)
                {
                        t= temp->next;
                        temp->next = t->next;
                        free(t);
                        last=temp;
                }
```

```
                else
                {
                        t = temp->next;
                        temp->next = t->next;
                        free(t);
                }
                printf("\n DELETED ");
                printf("\n The element in the list is : ");
                temp=last->next;
                while(temp!=last)
                {
                        printf("%d\t",temp->info);
                        temp=temp->next;
                }
                printf("%d\t",temp->info);
        }
        getch();
        break;
case 5: clrscr();
        printf("\n\nSearching an element");
        printf("\n\nPlease enter the element :");
        scanf("%d",&ele);
        flag=0;
        temp=last->next;
        while(temp!=last)
        {
                if(temp->info==ele)
                {
                        flag=1;
                        //printf("Present");
                }
                temp=temp->next;
        }
        if(last->info==ele)
        {
                printf("The element is present");
                flag=1;
        }
        if(flag==0)
        printf("The element is not present");
        else
        printf("\n\nThe element is  present");
        getch();
        break;
}//end switch
printf("\n \n\n\t 1. Continue");
printf("\t 2. Exit");
printf("\n \n Choice  : ");
scanf("%d",&c);
}while (c != 2);
```

```
        getch();
}
```

7. Double Link List

```
  /*Program for Inserting and Deleting element at front&middle using double linked  list. date
13.12.2005*/

#include<stdio.h>
#include<conio.h>
struct link
{
        int info;
        struct link *next,*prev;
};
struct link *t,*start,*temp;
void main()
{
        int ele,ch,i,pos,pos1,flag;
        start=NULL;
        while(1)
        {
                //clrscr();
                printf("\n\n\t\t 1.Insertion front \n\n\t\t 2.Deletion front");
                printf("\n\n\t\t 3.Insertion before \n\n\t\t 4.Deletion before");
                printf("\n\n\t\t 5.searching");
                printf("\n\n\t\t 6.display \n\n\t\t 7.Exit");
                printf("\n\n Enter your choice: ");
                scanf("%d",&ch);
                switch(ch)
                {
                case 1: /*Insert a node at front*/
                        t=(struct link *) malloc(sizeof(struct link));
                        printf("\n enter the element: ");
                        scanf("%d",&ele);
                        t->info=ele;
                        t->next=start;
                        start->prev=t;
                        start=t;
                        t=start;
                        while(t!=NULL)
                        {
                                printf(" ");
                                printf("\t %d",t->info);
                                t=t->next;
                        }
                        getch();
                        break;
                case 2: /*Deletion a node at front*/
                        t=start;
```

```
        start=start->next;
        t->next=NULL;
        start->next=NULL;
        if(t==NULL)
        {
                printf("\n\n\t\t Can't remove");
                getch();
                break;
        }
        printf("\n\n\t\t One Node %d removed",t->info);
        getch();
        free(t);
        break;
case 3: /*Insertion a node at middle*/
        printf("\n\n Enter the position: ");
        scanf("%d",&pos);
        temp=start;
        for(i=1;i<pos;i++)
        temp=temp->next;
        t=(struct link *) malloc(sizeof(struct link));
        printf("\n\n enter the element: ");
        scanf("\n\n %d",&ele);
        t->info=ele;
        t->next=temp->next;
        t->prev=temp;
        temp->next->prev=t;
        temp->next=t;
        t=start;
        while(t!=NULL)
        {
                printf(" ");
                printf("\t %d",t->info);
                t=t->next;
        }
        getch();
        break;
case 4: /* Deletion a node at middle*/
        printf("\n\n Enter the position: ");
        scanf("%d",&pos1);
        temp=start;
        for(i=1;i<pos1;i++)
        temp=temp->next;
        t=temp->next;
        temp->next=t->next;
        t->next->prev=temp;
        free(t);
        if(t==NULL)
        printf("Element can't inserted");
        break;
case 5: printf("\n\nSearching an element");
```

```
                        printf("\n_____");
                        printf("\n\nPlease enter the element :");
                        scanf("%d",&ele);
                        flag=0;
                        temp=start;
                        while(temp!=NULL)
                        {
                                if(temp->info==ele)
                                {
                                        printf("\n\nThe element is present");
                                        flag++;
                                }//end if
                                temp=temp->next;
                        }//end while
                        if(flag==0)
                        printf("\n\nThe element is not present");
                        getch();
                        break;
                case 6: /* Display*/
                        clrscr();
                        t=start;
                        while(t!=NULL)
                        {
                        printf(" ");
                        printf("\t %d",t->info);
                        t=t->next;
                        }
                        getch();
                        break;
                case 7: exit(0);
                        break;
                }
        }
}
```

8. Merge Short

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[10],x[10];
        int i,j,k,n,l1,u1,l2,u2,size;
        clrscr();
        printf("Enter how many element u want to sort:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\n Enter the %d Element:",i+1);
                scanf("%d",&a[i]);
```

```
        }
        size=1;
        while(size<n)
        {
                l1=0;
                k=0;
                while(l1+size<n)
                {
                        l2=l1+size;
                        u1=l2-1;
                        u2=(l2+size-1<n-1)? l2+size-1 : n-1;
                        //  u2=(l2+size-1<n)? l2+size-1 : n-1;
                        for(i=l1,j=l2;i<=u1&&j<=u2;k++)
                        {
                                if(a[i]<=a[j])
                                x[k]=a[i++];
                                else
                                x[k]=a[j++];
                        }//end for loop
                        for(;i<=u1;i++)
                        x[k++]=a[i];
                        for(;j<=u2;j++)
                        x[k++]=a[j];
                        l1=u2+1;
                }//end while
                for(i=l1;k<n;i++)
                x[k++]=a[i];
                for(i=0;i<n;i++)
                a[i]=x[i];
                size=size*2;
        }//end while
        printf("\n\nThe sorted numbers:");
        for(i=0;i<n;i++)
        printf("%3d",a[i]);
        getch();
}//end main
```

9. Heap Short

```
/* Heap sort */
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i,j,s,ele,f,x[10],temp;
        clrscr();
        printf("\n Enter the no of element:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
```

```
                printf("\nEnter the %d element:",i+1);
                scanf("%d",&x[i]);
        }
        for(i=0;i<n;i++)
        {
                ele=x[i];
                s=i;
                f=(s-1)/2;
                while(x[f]<ele&&s>0)
                {
                        x[s]=x[f];
                        s=f;
                        f=(s-1)/2;
                }//end while
                x[s]=ele;
        }
        for(i=n-1;i>=1;i--)
        {
                temp=x[i];
                x[i]=x[0];
                f=0;
                if(i==1)
                        s=-1;
                else
                        s=1;
                if(x[2]>x[1] && I > 2)
                        s=2;
                while( s > 0 && x[s] > temp)
                {
                        x[f]=x[s];
                        f=s;
                        s=2*f+1;
                        if(s+1<=i-1 && x[s+1] > x[s])
                         s=s+1;
                        if( s > i-1)
                                s = -1;
                }
                x[f]=temp;
        }
        printf("\n\nThe sorted list:");
        for(i=0;i<n;i++)
        printf("\t%d",x[i]);
        getch();
 }
```

10. Quick Short

```
/*QUICK SORT*/
```

The Quick Sort

It works as follows: It places a number in its correct position in the list and then recursively uses the same scheme for the group of numbers below the correctly positioned number and then for the group above it. On the average, the quick sort is the most efficient sort. Let's examine the list 16 4 11 9 71 29 8 13 stored in the arrray `X` to see how it works.

Call the first number (here, 16) `first`; the subscript 1, `beginning`; and the subscript of the last element `ending`. The procedure will use the heading:

```
PROCEDURE Quick(Beginning, Ending:integer; VAR X : Sorted);
```

We'll place `first` in a location such that the numbers to the right of it are greater than it, and the numbers to the left of it will be less than it. In other words, it will be in its correct position in the list. Do this by going forward through the list until you find a number (here, 71) greater than or equal to `first`. Then go from the back of the list to the left until we find a number (here, 13) that's less than or equal to `first`. If the subscript (I) of the first number (71) is less than the subscript (J) of the second number (13) as is the case here, exchange the two numbers. The list is now 16 4 11 9 13 29 8 71.

Continue by preceding to the right from the position that 13 is in until you find the next number (here, 29) greater than or equal to `first`. Then go to the left from 71 until you find the next number (here, 8) less than or equal to `first`. If the subscript (I) of the first number (29) is less than that (J) of the second number (8) as is the case here, exchange the two numbers. The list is now 16 4 11 9 13 8 29 71.

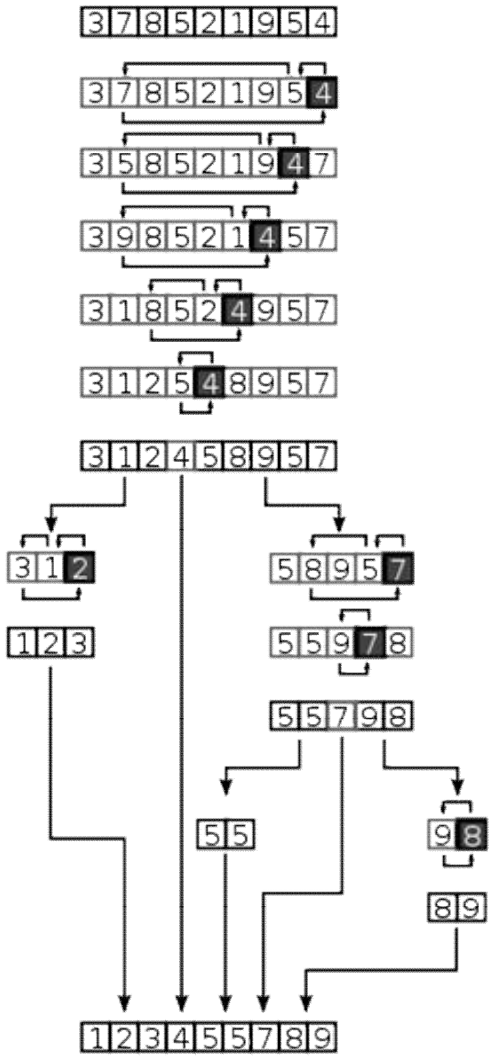
At this point, the value of I is 6 and that of J is 7. `X[I]` is 8 and `X[J]` is 29. Since `X[I] <= first`, increment I to 7, and since `X[J] > first`, decrement J to 6. Now $I > J$. This is called *cross over*. When cross over occurs, exchange `X[J]` with `first`. In our case, exchange 16 with 8. The list is now 8 4 11 9 13 16 29 71 `first`, 16, is now in its proper position in the list.

We then have to apply this technique to the group of numbers to the right of `first` and to the group to its left (this is the recursion part), i., e., call `Quick(J + 1, Ending, X)` and `Quick(Beginning, J - 1, X)`. These two groups are shown in square brackets [8 4 11 9 13] 16 [29 71]. For the [29, 71] sublist, cross over occurs immediately, and J becomes 1. So the switch is the identity operation and nothing is changed. `Quick(J + 1, Ending, X)` becomes `Quick(2, 2, X)` and `Quick(Beginning, J - 1, X)` becomes `Quick(1, 1, X)`. In both cases `Quick` is sorting only one element. In order to have the procedure return immediately, place `if I < J then` at the beginning of the procedure.

At this point, the procedure `Quick` should be clear.

```
#include<stdio.h>
#include<conio.h>
void quicksort(int[],int,int);
void main()
{
        int x[10],ele,n,i,lb,ub;
        clrscr();
        printf("\n Enter the no element:");
        scanf("%d",&n);
        lb=0;
        ub=n-1;
        for(i=0;i<n;i++)
        {
                printf("\n Enter the %d element:",i+1);
                scanf("%d",&x[i]);
        }
        quicksort(x,lb,ub);
        // clrscr();
        printf("\n\n\nSorted list:");
        for(i=0;i<n;i++)
        printf("\t%d",x[i]);
        getch();
}
 void quicksort(int x[],int lb,int ub)
 {
        int a,i,t,up,down,j;
        if(lb>=ub)
        return;
        down=lb;
        up=ub;
        a=x[lb];
        while(down<up)
        {
                while(x[down]<=a&&down<up)  down++;
                while(x[up]>a)  up--;
                if(down<up)
                {
                        t=x[down];
                        x[down]=x[up];
                        x[up]=t;
                }//end if
        }//end while
        x[lb]=x[up];
        x[up]=a;
        j=up;
        quicksort(x,lb,j-1);
        quicksort(x,j+1,ub);
 }//end function
```

```
1.  //Quicksort w/o a separate compare function :)
2.  void quicksort(int A[], int lo, int hi) {
3.  int i, j, pivot, temp;
4.
5.  if(lo == hi) return;
6.  i=lo;
7.  j=hi;
8.  pivot= A[(lo+hi)/2];
9.
10. /* Split the array into two parts */
11. do {
12. while (A[i] < pivot) i++;
13. while (A[j] > pivot) j--;
14. if (i<=j) {
15. temp= A[i];
16. A[i]= A[j];
17. A[j]=temp;
18. i++;
19. j--;
20. }
21. } while (i<=j);
22.
23. if (lo < j) quicksort(A, lo, j);
24. if (i < hi) quicksort(A, i, hi);
25. }
```



11. Binary Tree

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
struct bstree
{
        int data;
        struct bstree *left;
        struct bstree *right;
};
struct bstree* insertbst(struct bstree*,int);
struct bstree* remove_elt(struct bstree*,int);
// struct bstree *root,*temp,*t;
void main()
```

```
{
        struct bstree *root,*temp,*t;
        struct bstree *new;
        int option,ele,info;
        root=NULL;
        clrscr();
        while(1)
        {
                printf("\n\n1.insert\n\n2.delete\n\n3.display\n\n4.exit");
                printf("\n\nEnter the choice:");
                scanf("%d",&option);
                switch(option)
                {
                case 1: printf("\n\nEnter the element to insert:");
                        scanf("%d",&ele);
                        root=insertbst(root,ele);
                        break;
                case 2: printf("Enter the element to remove:");
                        scanf("%d",&ele);
                        root=remove_elt(root,ele);
                        break;
                case 3: if(root==NULL)
                        printf("Tree is empty");
                        else
                        print(root);
                        break;
                case 4: exit(0)
                }
        }
}
struct bstree* insertbst(struct bstree *root,int x)
{
        if(root==NULL)
        {
                root=(struct bstree *) malloc(sizeof(struct bstree));
                root->data=x;
                root->left=root->right=NULL;
        }
        else if(x<=root->data)
                 root->left=insertbst(root->left,x);
        else if(x>root->data)
                root->right=insertbst(root->right,x);
        return root;
}
struct bstree* findmin(struct bstree *t)
{
        if(t==NULL)
        return NULL;
        else
        if(t->left !=NULL)
```

```
        return findmin(t->left);
        else
        return t;
}
struct bstree* remove_elt(struct bstree *t, int key)
{
        struct bstree *temp;
        if(t==NULL)
        printf("node is not available");
        else if(key < t->data)
        t->left=remove_elt(t->left,key);
        else if(key > t->data)
        t->right=remove_elt(t->right,key);
        else if((t->left !=NULL)&&(t->right!=NULL))
        {
                temp=findmin(t->right);
                t->data=temp->data;
                t->right=remove_elt(t->right,t->data);
        }
        else
        {
                temp=t;
                if(t->left==NULL)
                t=t->right;
                else if(t->right==NULL)
                t=t->left;
                free(temp);
        }
        return t;
 }
 print(struct bstree *root)
 {
        if(root!=NULL)
        {
                if(root->left !=NULL)
                print(root->left);
                printf("\n%d",root->data);
                if(root->right!=NULL)
                print(root->right);
        }
        return;
}
```

12. Tree Traversal

```
#include<stdio.h>
#include<conio.h>
struct node
{
        int element;
        struct node *left;
```

```
        struct node *right;
};
typedef struct node tree;
tree *insert(char,tree*);
void inorder(tree *);
void postorder(tree*);
void preorder(tree*);
void main()
{
        int ele,j;
        int ch;
        tree *t;
        t=NULL;
        j=0;
        do
        {
                clrscr();
                printf("\n\nTree menu\n");
                printf("\n\nInsertion ********>1");
                printf("\n\nInorder traversal ********>2");
                printf("\n\nPostorder traversal*******>3");
                printf("\n\npreorder traversal*******>4");
                printf("\n\nExit**********>5");
                printf("\n\nEnter the choice :");
                //printf("\n");
                scanf("%d",&ch);
                switch(ch)
                {
                case 1: printf("\n\nEnter the element");
                        scanf("%d",&ele);
                        t=insert(ele,t);
                        printf("%d",t);
                        break;
                case 2: inorder(t);
                        getch();
                        break;
                case 3: postorder(t);
                        getch();
                        break;
                case 4: preorder(t);
                        getch();
                        break;
                case 5: //printf("\n-----exit");
                        getch();
                        exit(1);
                }
                j++;
        }while(j<18);
}
tree *insert(char x,tree* t)
```

```
{
        if(t==NULL)
        {
                t=(tree *) malloc(sizeof(tree));
                t->element=x;
                t->left=NULL;
                t->right=NULL;
        }
        else if(x<t->element)
        t->left=insert(x,t->left);
        else if(x>t->element)
        t->right=insert(x,t->right);
        return t;
}
void inorder(tree *t)
{
        if(t!=NULL)
        {
                inorder(t->left);
                printf("\t%d",t->element);
                inorder(t->right);
        }
        else
                return;
}
void preorder(tree *t)
{
        if(t!=NULL)
        {
                printf("\t\t%d",t->element);
                preorder(t->left);
                preorder(t->right);
        }
        else
                return ;
}
void postorder(tree *t)
{
        if(t!=NULL)
        {
                postorder(t->left);
                postorder(t->right);
                printf("\t\t%d",t->element);
        }
        else
                return ;
 }
```

13. Search

/* Depth first search and breadth first search */

```
#include<stdio.h>
#include<conio.h>
int a[10][10],visited[10],n;
void main()
{
	int i,j;
	void dfs(int);
	void bfs(int);
	clrscr();
	printf("Enter the number of nodes :");
		scanf("%d",&n);
	for(i=1;i<=n;i++)
		visited[i]=0;
	printf("Enter the adjacent nodes \n\n");
	for(i=1;i<=n;i++)
	for(j=1;j<=n;j++)
		if(i!=j && i<j)
		{
			printf("Is node[%d] node[%d]  adjacent ? :",i,j);
			scanf("%d",&a[i][j]);
		}
	printf("Depth first traversl :\n\n");
	for(i=1;i<=n;i++)
		if (visited[i]==0)
			dfs(i);
	for(i=1;i<=n;i++)
	{
		visited[i]=0;
	}
	printf("\n\nBreadth first traversal :\n\n");
	for(j=1;j<=n;j++)
	{
		if (visited[j]==0)
			bfs(j);
	}
	getch();
}
void dfs(int node)
{
	int i;
	printf("%d  ",node);
	visited[node]=1;
	for(i=1;i<=n;i++)
		if (visited[i]==0)
			if(a[node][i]!=0)
				dfs(i);
}
void bfs(int node)
{
```

```
        int i,j;
        printf("%d  ",node);
        visited[node]=1;
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
                if (visited[j]==0)
                        if(a[i][j]!=0)
                                bfs(j);
}
```