Objective : 1
   To find the sum of individual digits of a given number

Description:
   Sum of the individual digits means adding all the digits of a number
   Ex: 123  sum of digits is 1+2+3=6

Algorithm:

   Step 1: start
   Step 2: read n
   Step 3: initialize the s=0
   Step 4: if n<0 goto Step 7
   Step 5: if n!=0 goto Step 6 else goto step 7
   Step 6: store n%10 value in p
      Add p value to s
      Assign n/10 value to n
      Goto Step 5
   Step 7: print the output
   Step 8:stop
Flowchart:

START

S=0

READ N

IF N<0

OUTPUT

STOP

TRUE

WHILE N! =0

TRUE

P=N%10

S=S+P

N=N/10

FALSE
**Program:**
#include<stdio.h>
main()
{
    int n,s,p;
    clrscr();
    printf("enter the vaue for n:\n");
    scanf("%d",&n);
    s=0;
    if(n<0)
    printf("The given number is not valid");
    else
    {
        while(n!=0)     /* check the given value =0 or not */
        {
            p=n%10;
            n=n/10;
            s=s+p;
        }
        printf("sum of individual digits is %d",s);
    }
    getch();
}

**Output:**
1. Enter the value for n: 333
   Sum of individual digits is 9
2. Enter the value for n: 4733
   Sum of individual digits is 17
3. Enter the value for n: -111
   The given number is not valid

**Conclusion:** The program is error free

**VIVA QUESATIONS:**

1) What is the mean of sum of the individual digits?
   Ans: Sum of the individual digits means adding each digit in a number

2) What is positive integer?
   Ans: If the integer value is greater than zero then it is called positive integer

3) Define preprocessor?
   Ans: Before compiling a process called preprocessing is done on the source code by a program called the preprocessor.
**Objective:**
To print the Fibonacci series for 1 to n value

**Description**
A Fibonacci series is defined as follows
- The first term in the sequence is 0
- The second term in the sequence is 1
- The subsequent terms are found by adding the preceding two terms in the sequence

Formula: let $t_1, t_2, \ldots, t_n$ be terms in Fibonacci sequence

- $t_1 = 0, t_2 = 1$
- $t_n = t_{n-2} + t_{n-1}$ where $n > 2$

**Algorithm:**

Step 1: start
Step 2: initialize $a = 0, b = 1$
Step 3: read $n$
Step 4: if $n == 1$ print $a$ go to step 7, else goto step 5
Step 5: if $n == 2$ print $a, b$ go to step 7 else print $a, b$
Step 6: initialize $i = 3$
  i) if $i <= n$ do as follows. If not goto step 7
     c = $a + b$
     print $c$
     $a = b$
     $b = c$
     increment $i$ value
     goto step 6(i)

Step 7: stop
Flowchart:

START

A=0, b=1

Read n

If n == 1
  Output a

If n == 2
  Output a, b

I=2
i++

I<n

True

If n == 1

Output a

False

If n == 2

Output a, b

Stop

True

C = a + b

Output c

A = b
B = c

False

Stop
Program:

#include<stdio.h>
void main()
{
    int a,b,c,n,i;
    clrscr();
    printf("enter n value");
    scanf("%d",&n);
    a=0;
    b=1;
    if(n==1)
        printf("%d",a);
    else
        if(n==2)
            printf("%d%d",a,b);
        else
        {
            printf("%d%d",a,b);
            //LOOP WILL RUN FOR 2 TIME LESS IN SERIES AS THESE WAS PRINTED IN ADVANCE
            for(i=2;i<n;i++)
            {
                c=a+b;
                printf("%d",c);
                a=b;
                b=c;
            }
        }
    getch();
}

Output:
1. Enter n value : 5
   0 1 1 2 3
2. Enter n value : 7
   0 1 1 2 3 5 8
3. Enter n value : -6
   0 1

Conclusion: The program is error free
VIVA QUESTIONS:

1) **What is Fibonacci series ?**
   Ans: A fibonacci series is defined as follows
   The first term in the sequence is 0
   The second term in the sequence is 1
   The subsequent terms 1 found by adding the preceding two terms in the sequence
   Formula: let t1,t2,.........tn be terms in fibinacci sequence
   t1=0, t2=1
   tn=tn-2+tn-1……where n>2

2) What are the various types of unconditional statements?
   Ans: goto,Break and continue

3) What are the various types of conditional statements?
   Ans: if , if else ,switch statements

4) Expand <STDIO.H>?
   Ans: standard input output header file
Objective: 3

To print prime numbers up to 1 to n

Description:
Prime number is a number which is exactly divisible by one and itself only
Ex: 2, 3, 5, 7, ………;

Algorithm:
Step 1: start
Step 2: read n
Step 3: initialize i=1, c=0
Step 4: if i<=n goto step 5
    If not goto step 10
Step 5: initialize j=1
Step 6: if j<=1 do as the follow. If no goto step 7
    i) if i%j==0 increment c
    ii) increment j
    iii) goto step 6
Step 7: if c==2 print i
Step 8: increment i
Step 9: goto step 4
Step 10: stop
Flow chart:

Start

Read n

I = 1
I <= n
I++

J = 1
J <= i
J++

If fact == 2

If I % j == 0
Fact ++

true

Output fact

false

false

false

false

If I % j == 0

true

Fact ++

false

true

false

stop

false
Program:
#include<stdio.h>
#include<conio.h>
void main()
{
    int n,i,fact,j;
    clrscr();
    printf("enter the number:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        fact=0;
        //THIS LOOP WILL CHECK A NO TO BE PRIME NO. OR NOT.
        for(j=1;j<=i;j++)
        {
            if(i%j==0)
            fact++;
        }
        if(fact==2)
        printf("\n%d",i);
    }
    getch( );
}

Output:
Enter the number : 5
    2 3 5
Enter the number : 10
    2 3 5 7
Enter the number : 12
    2 3 5 7 11

Conclusion : The program is error free

Viva Questions:
1) What is prime number ?
Ans: Prime number is a number which is exactly divisible by one and itself only

2)What is an algorithm?
Ans : A step by step procedure is called algorithm

3)What is flow chart?
Ans: A pictorial representation an algorithm is called a flow chart
4)What is program?
Ans : A collection of statements is called
Objective: To find the roots of the quadratic equation

Description:
Nature of roots of quadratic equation can be known from the quadrant \(\Delta = b^2 - 4ac\)
If \(b^2 - 4ac > 0\) then roots are real and unequal
If \(b^2 - 4ac = 0\) then roots are real and equal
If \(b^2 - 4ac < 0\) then roots are imaginary

Algorithm:
Step 1: start
Step 2: read the a, b, c value
Step 3: if \(b^2 - 4ac > 0\) then
   Root 1 = \((-b + \text{pow}((b^2 - 4ac), 0.5))/2a\)
   Root 2 = \((-b - \text{pow}((b^2 - 4ac), 0.5))/2a\)
Step 4: if \(b^2 - 4ac = 0\) then
   Root1 = Root2 = \(-b/(2a)\)
Step 6: print roots
Step 7: stop
Flowchart:

Start

Read a, b, c

D = pow(b*b-4*a*c), 0.5

If d > 0

R1 = ((-b+D) / (2*a))
R2 = ((-b-D) / (2*a))

Output R1, R2

Stop

If d == 0

R1 = -b / (2 * a)
R2 = -b / (2 * a)

Imaginary roots
Program:

```c
#include<stdio.h>
#include<math.h>
void main()
{
    float a,b,c,r1,r2,d;
    clrscr();
    printf("Enter the values for equation:");
    scanf("%f%f%f",&a,&b,&c);
    /* check the condition */
    if(a==0)
        printf("Enter value should not be zero ");
    else
    {
        d=b*b-4*a*c;
        /* check the condition */
        if(d>0)
        {
            r1=(-b+sqrt(d)/(2*a));
            r2=(-b-sqrt(d)/(2*a));
            printf("roots are real and unequal\n");
            printf("%f\n%f\n",r1,r2);
        }
        else
        if(d==0)
        {
            r1=-b/(2*a);
            r2=-b/(2*a);
            printf("roots are real and equal\n");
            printf("root=%f\n",r1);
            printf("root=%f\n",r2);
        }
        else
            printf("roots are imaginary");
    }
    getch();
}
```

Output:
1. Enter the values for equation: 1, 6, 9
   Roots are real and equal
   Root= -3.0000
   Root= -3.0000
2. Enter the values for equation: 2, 7, 6
   Roots are real and unequal
   Root= -6.75
   Root= -7.25
3. Enter the values for equation: 1, 2, 3
   Roots are imaginary

Conclusion: The program is error free

VIVA QUESATIONS:

1) What are various types of loop statements?
   Ans: While, do-while, for loop statements

5) What is the difference between while and do-while statements?
   Ans: In while the condition will be checked first and then enter into a loop.
       But in do-while the statements will be executed first and then finally check the
       Condition.

3) How to find the roots of quadratic equations?
   Ans: Nature of roots of quadratic equation can be known from the quadrant
       \[ \Delta = b^2 - 4ac \]
       If \( b^2 - 4ac > 0 \) then roots are real and unequal
       If \( b^2 - 4ac = 0 \) then roots are real and equal
       If \( b^2 - 4ac < 0 \) then roots are imaginary

4) List out the C features?
   Ans: Portability, flexibility, wide acceptability etc.,
**Objective:**

To calculate the sum. Sum=\(-x^2/2!+ x^4/4!- x^6/6!+ x^8/8!- x^{10}/10!\)

**Algorithm:** main program:

Step 1: start
Step 2: declare x,i,n,s=0,c
Step 3: read x value
Step 4: for i=0 , n=0; i<=10; i=i+2, n++ goto step 5
Step 5: s=s+(\((-1)^n x^i/\text{fact}(i)\))
Step 6: print s value
Step 7: stop

**Sub program:**

Step 1: while x!=0 goto Step 2
Step 2: y=y+x; x—
Step 3: return y
Step 4: return to main program
Flowchart:

Start

Read x, I, n, c

S = 0

I = 0, n = 0, i <= 10

I = I + 2, n++

S = S + (pow(-1, n) * pow(x, i) / fact(i))

Print S

Stop

Sub Program

Fact()

While x! = 0

Y = y * x
x--

Return y

Return to main program
Program:
#include<stdio.h>
#include<math.h>
long fact(int);
void main()
{
    int x,i,n;
    float s=0,c;
    clrscr();
    printf("\n enter the value of x\t");
    scanf("%d",&x);
    /*perform the looping operation*/
    for(i=0,n=0;i<=10;i=i+2,n++)
        s=s+(pow(-1,n)*pow(x,i)/fact(i));
    printf("\n the result is %f",s);
    getch();
}
/* calling sub program*/
long fact(int x)
{
    long int y=1;
    while(x!=0)
    {
        y=y*x;
        x--;
    }
    return y;
}

Output:
1.Enter the value of x : 1
    The result is 0.540302
2 Enter the value of x: 2
    The result is -0.416155

Conclusion: The program is error free

VIVA QUESATIONS:

1)  What is function ?
    Ans: A function is a sub program it returns a value.

2)  What is procedure ?
    Ans: A procedure is a sub program it does not returns a value

3)  What are the basic data types in C ?
    Ans: int, char, float, double

4)  How to define preprocessor ?
    Ans: By using the # symbal Ex: #include<stdio.h>
**Objective:** 6

Programs that use recursive function to find the factorial of a given integer.

**Description:**
Factorial of a number is nothing but the multiplication of numbers from a given number to 1

**Algorithm:** main program
- Step 1: start
- Step 2: read n
- Step 3: call sub program as f=fact(n)
- Step 4: print f value
- Step 5: stop

**Sub program:**
- Step 1: initialize the f
- Step 2: if n= = 0 or n == 1 return 1 to main program if not goto step 3
- Step 3: return n*fact(n-1) to main program
Flowchart:

Start

Read n

Call subprogram
\( F = \text{fact}(n) \)

output

Stop

Sub program

Fact ()

If \( n=0 \) \(\text{or}\) \( n=1 \)

false

true

Return \( n*\text{fact}(n-1) \)

Return to main program
**Program:**

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

int fact(int n)
{
    int f;
    if((n==0)||(n==1)) // check the condition for the n value
        return(n);
    else
        f=n*fact(n-1);   //calculate the factorial of n
    return(f);
}

void main()
{
    int n;
    clrscr();
    printf("enter the number : ");
    scanf("%d",&n);
    printf("factorial of number%d",fact(n));
    getch();
}
```

**Output:**
1. Enter the number : 5
   Factorial of number: 120
2. Enter the number : 3
   Factorial of number: 6
3. Enter the number : 9
   Factorial of number: -30336

**Conclusion:** the program is error free

**VIVA QUESATIONS:**

1) What is the meaning of factorial number?
**Ans:** Factorial of a number is nothing but the multiplication of numbers from a given number to 1

2) What is the meaning of recursive function?
**Ans:** A function call itself is called recursive function

3) Define library functions?
**Ans:** The functions have already been written, compiled and placed in libraries and are called library functions.

4) Define formal parameters?
**Ans:** Formal parameters are the parameters given in the function declaration as function definition.
Objective: 7

Program that use non recursive function to find the factorial of a given integer.

Description:
Factorial of a number is nothing but the multiplication of numbers from a given number to 1
Ex: 5! = 5*4*3*2*1 = 120

Algorithm: main program
Step 1: start
Step 2: read n
Step 3: call the sub program fact(n)
Step 4: print the f value
Step 5: stop

Sub program:
Step 1: initialize the f = 1
Step 2: if n == 0 or n = 1 return 1 to main program. If not goto step 3
Step 3: perform the looping operation as follows
    For i = 1 i <= n; i++
Step 4: f = f * i
Step 5: return f value to the main program
Flowchart:

**Factorial nonrecursive**

- **start**
- Read i
- Call subprogram Fact(n)
- Print output Value of fact
- Stop

**Sub program**

Fact ( )

- F = 1, i

If \( n == 0 \) || \( n == 1 \)

- I = 1
- I <= n
- I++
- \( F = f \times i \)

Return to main program


**Program:**

```c
#include<stdio.h>
#include<conio.h>
int fact(int n) //starting of the sub program
{
    int f=1,i;
    if((n==0)||(n==1)) // check the condition for n value
        return(1);
    else
        for(i=1;i<=n;i++) // perform the looping operation for calculating the factorial
            f=f*i;
        return(f);
}
void main()
{
    int n;
    clrscr();
    printf("enter the number : ");
    scanf("%d", &n);
    printf("factorial of number%d",fact(n));
    getch();
}
```

**Output:**
1. Enter the number: 7
   Factorial of number: 5040
2. Enter the number: 6
   Factorial of number: 720
3. Enter the number: 8
   Factorial of number: -25216

**Conclusion:**
The program is error free

**VIVA QUESATIONS:**
1) What is meant by call by value ?
   Ans: passing values to the function as arguments
2) What is meant by call by reference ?
   Ans: passing address to the function as arguments
3) define actual parameters ?
   Ans: The actual parameters often known as arguments are specified in the function call.
**Objective: 8**
To find the GCD of two given integers by using the recursive function

**Description:**
GCD means Greatest Common Divisor. i.e the highest number which divides the given number
Ex: GCD(12,24) is 12
Formula: GCD= product of numbers/ LCM of numbers

**Algorithm:** main program
Step 1: start
Step 2: read a,b
Step 3: call the sub program GCD(a,b) for print the value
Step 4: stop

**Sub program:**
Step 1: if n>m return GCD(n,m)
Step 2: if n==0 return m else goto step 3
Step 3: return GCD (n,m%n)
Step 4: return to main program
Flowchart:

Start

Read a, b

Call sub program
G = gcd(a, b)

Print gcd value

Stop

Gcd( )

false

If n > m

true

If n == 0

Return m

Call the same function
Return gcd(n, m%dn)

Return to main program

Return Gcd(n, m)
**Program:**

```c
#include<stdio.h>
#include<conio.h>
int gcdrecursive(int m,int n) // starting of the sub program
{
    if(n>m)
        return gcdrecursive(n,m);
    if(n==0)
        return m;
    else
        return gcdrecursive(n,m%n); // return to the main program
}
void main()
{
    int a,b,igcd;
    clrscr();
    printf("enter the two numbers whose gcd is to be found:");
    scanf("%d%d",&a,&b);
    printf("GCD of a,b is  %d",gcdrecursive(a,b)); // return to the sub program
    getch();
}
```

**Output:**
1. enter the two numbers whose gcd is to be found:5,25
   GCD of a,b is : 5
2. enter the two numbers whose gcd is to be found:36,54
   GCD of a,b is : 18
3. enter the two numbers whose gcd is to be found:11,13
   GCD of a,b is : 1

**Conclusion:**
The program is error free

**VIVA QUESTIONS:**

1) What is meaning of GCD ?
   Ans: GCD means Greatest Common Divisor. i.e the highest number which divides the given number

2) Define scope of a variable ?
   Ans: The scope of a variable can be define as the region over which the variable is accessible

3) Show an scope resolution operator ?
   Ans: double colon(::)

4) Define extent of a variable ?
   Ans: The period of time during which memory is associated with a variable is called extent of the variable.
**Objective:** 9
To find the GCD of two given integers by using the non recursive function

**Description:**
GCD means Greatest Common Divisor. i.e the highest number which divides the given number
Ex: GCD(12,24) is 12
Formula: GCD= product of numbers / LCM of numbers

**Algorithm:**
Step 1: start
Step 2: read a,b
Step 3: call sub program g=GCD(a,b)
Step 4: print the g value
Step 5: stop

Sub program:
Step 1: initialize the p=1, q, remainder
Step 2: remainder=p-(p/q*q)
Step 3: remainder=0 return q else goto step 4
Step 4: GCD(q,remainder) return to main program
Flowchart:

1. Start
2. Read a, b
3. Call subprogram g=gcd(a, b)
4. Output
5. Stop

Gcd()

Remainder = p - (p/q*q)

If remainder == 0
   Gcd(q, remainder)
   Return to main program
Else
   Return q
Program:
#include<stdio.h>
#include<conio.h>
#include<math.h>
int gcdnonrecursive(int m,int n)
{
    int remainder;
    remainder=m-(m/n*n);
    if(remainder==0)
        return n;
    else
        gcdnonrecursive(n,remainder);
}
void main()
{
    int a,b,igcd;
    clrscr();
    printf("enter the two numbers whose gcd is to be found:");
    scanf("%d%d",&a,&b);
    printf("GCD of %d",gcdnonrecursive(a,b));
    getch();
}

Output:
1. enter the two numbers whose gcd is to be found:5,25
   GCD of a,b is : 5
2. enter the two numbers whose gcd is to be found:36,54
   GCD of a,b is : 18
3. enter the two numbers whose gcd is to be found:11,13
   GCD of a,b is : 1

Conclusion:
The program is error free

VIVA QUEATIONS:
1)What is meaning of GCD ?
   Ans:
       GCD means Greatest Common Divisor. i.e the highest number which divides the given number
Objective: To solve the towers of Hanoi problem by using the recursive function

Description:
Towers of Hanoi problem means we have three towers

Here source, intermediate, and destination are the three towers. We have to transfer all the disks from source to destination towers. Here the restriction is not to place a big disk on a smaller one. For this, we use intermediate tower. Finally, the arrangements in the destination tower must be the same as the disks in the source tower at first.

Algorithm: main program

Step 1: start
Step 2: initialize the source=a, intermediate=c, destination=d
Step 3: read n
Step 4: call the sub program Hanoi recursion (n value, a, b, c)
Step 5: stop

Sub program:
Step 1: if n==1 call the sub program Hanoi recursion (num-1, a, c, b)
Step 2: print the output from a to b
Step 3: call the sub program Hanoi recursion (num-1, b, c, a)
Step 4: return to main program
Flowchart:

START

SOURCE = A
INTERMEDIATE = C
DESTINATION = B

READ n

Call subprogram
Hanoi(num,source,intermediate,destination) A, C, B

Stop

hanoirecursive( )

false

true

If num==1

Call subprogram
Num-1, source "A"

Print A, C

Print A, C

Call subprogram
Hanoi(num-1, B, C, A))

Print A, C

Return to main program
Program:
#include<stdio.h>
#include<conio.h>
void Hanoirecursion(int num,char ndl1,char ndl2,char ndl3)
{
    if(num==1)
    {
        printf("Move top disk from needle %c to needle %c",ndl1,ndl2);
        return;
    }
    Hanoirecursion(num-1,ndl1,ndl3,ndl2);
    printf("Move top disk from needle %c to needle %c",ndl1,ndl2);
    Hanoirecursion(num-1,ndl3,ndl2,ndl1);
}
void main()
{
    int no;
    clrscr();
    printf("Enter the no. of disk to be transferred:");
    scanf("%d",&no);
    if(no<1)
        printf("\n There's nothing to move");
    else
        printf("\n recursive");
    Hanoirecursion(no,'A','B','C');
    getch();
}

Outputs:
1. Enter the no. of disk to be transferred :3
   Move top disk from needle a to needle b
   Move top disk from needle a to needle c
   Move top disk from needle b to needle c
   Move top disk from needle a to needle b
   Move top disk from needle c to needle a
   Move top disk from needle c to needle b
   Move top disk from needle a to needle b

Conclusion:
The program is error free

VIVA QUESTIONS:
1) What is purpose of towers of Hanoi?
   Ans: We have to transfer all the disks from source to destination towers. Here the restriction is not to place a big disk on smaller one. for this we use intermediate tower. Finally the arrangements in the destination tower must be as same as the disks in the source tower at first.
**Objective:**
To solve the towers of Hanoi problem by using the non recursive function

**Description:**
Towers of Hanoi problem means we have three towers

Here source, intermediate and destination are the three towers. We have to transfer all the disks from source to destination towers. Here the restriction is not to place a big disk on smaller one. For this we use intermediate tower. Finally, the arrangements in the destination tower must be as same as the disks in the source tower at first.

**Algorithm:**

Step 1: start
Step 2: declare the no
Step 3: read the no value
Step 4: if (no<1) Print nothing to move
    Else Print nonrecursion
Step 5:Hanoi non recursion(no,,A’,’B’,’C’)
Step 6:stop
Sub program:
Step 1: Declare num,sndl,indl,dndl,
    stk[ ],stksndl[ ],stkendl[ ],stkadd[ ],
    temp,top,add
Step 2: declare the top=NULL
Step 3: one:
    If(num==1)then
        Print the output value
        Goto four
Step 4: two:
    Top=top+1
    Stkn[top]=num
    Stksndl[top]=sndl
    Stkindl[top]=indl
    Stkdndl[top]=dndl
    Stkadd[top]=3
    Num=num-1
Sndl=sndl
Temp=indl
Indl=dndl
Dndl=temp
Goto one. Goto step 3

Step 5:
  Three:
    Print the output
    Top=top+1
    Stkn[top]=num
    Stksndl[top]=sndl
    Stkindl[top]=indl
    Stkdndl[top]=dndl
    Stkadd[top]=5
    Num=num-1
    temp=sndl
    sndl=indl
    Indl=temp
    Dndl=dndl
    Goto one. Goto step 3

Step 6:
  Four:
    If(top==NULL)
    Return to main program
    Num= stkn[top]
    Sndl= stksndl[top]
    Indl= stkindl[top]
    Dndl=stkdndl[top]
    Add=stkadd[top]
    Top=top-1
    If(add==3)
    Goto three. Goto step 5
    Else
    If(add==5)

Step 7: return to main program
Flow chart:

Start

Read no

If no< 1

false

true

Print nothing to move

Hanoi nonrecursion
(no,.A',B',C')

Stop

Print nonrecursion
Hanoi nonrecursion ( )

Declare num, sndl, indl, dndl, stkn[], stksndl[], stkindl[], stkndnl[], stkadd[], temp, top, add

Top = NULL

If num==1

Print the value

If top=NULL

Num = stkn[top]
Sndl = stksndl[top]
Indl = stkindl[top]
Dndl = stkndnl[top]
Add = stkadd[top]
Top = top-1

If add==3

false

If add==5

one

Print value

Top = top+1
Num = num-1
temp = sndl
sndl = indl
Indl = temp
Dndl = dndl
Top = top+1

two

four

If top==NULL

Num = stkn[top]
Sndl = stksndl[top]
Indl = stkindl[top]
Dndl = stkndnl[top]
Add = stkadd[top]
Top = top-1

three
Program:
#include<stdio.h>
#include<conio.h>
void Hanoinonrecursion(int num,char sndl,char indl,char dndl)
{
    char stkn[100],stksndl[100],stkindl[100],stkdndl[100],stkadd[100],temp;
    int top,add;
    top=NULL;
    one:
        if(num==1)
            {
                printf("\n Move top disk from needle %c to needle %c",sndl,dndl);
                goto four;
            }
    two:
        top=top+1;
        stkn[top]=num;
        stksndl[top]=sndl;
        stkindl[top]=indl;
        stkdndl[top]=dndl;
        stkadd[top]=3;
        num=num-1;
        sndl=sndl;
        temp=indl;
        indl=dndl;
        dndl=temp;
        goto one;
    three:
        printf("\n Move top disk from needle %c to needle %c",sndl,dndl);
        top=top+1;
        stkn[top]=num;
        stksndl[top]=sndl;
        stkindl[top]=indl;
        stkdndl[top]=dndl;
        stkadd[top]=5;
        num=num-1;
        temp=sndl;
        sndl=indl;
        indl=temp;
        dndl=dndl;
        goto one;
    four:
        if(top==NULL)
            return;
return;
num=stkn[top];
sndl=stksndl[top];
indl=stkindl[top];
dndl=stkdndl[top];
add=stkadd[top];
top=top-1;
if(add==3)
goto three;
else if(add==5)
goto four;
}

void main()
{
    int no;
    clrscr();
    printf("Enter the no. of diss to be transferred:");
    scanf("%d",&no);
    if(no<1)
        printf("n There's nothing to move");
    else
        printf("n nonrecursive");
        Hanoinonrecursion(no,'A','B','C');
    getch();
}

Output:
1. Enter the no. of diss to be transferred:3
   nonrecursive
   Move top disk from needle A to needle C
   Move top disk from needle A to needle B
   Move top disk from needle C to needle B
   Move top disk from needle A to needle C
   Move top disk from needle B to needle A
   Move top disk from needle B to needle C
   Move top disk from needle A to needle C
   Conclusion: The program is error freed

VIVA QUESATIONS:
1) What is purpose of towers of Hanoi ?
   Ans: We have to transfer all the disks from source to destination towers. Here the
   restriction is not to place a big disk on smaller one . for this we use intermediate
tower. Finally the arrangements in the destination tower must be as same as the
disks in the source tower at first.

2) What is an array ?
   Ans: An array is a sequence of memory location of same data type.
**Objective: 12**

The total distance travelled by vehicle in 't' seconds is given by distance = \( ut + \frac{1}{2}at^2 \) where 'u' and 'a' are the initial velocity (m/sec.) and acceleration (m/sec2). Write C program to find the distance travelled at regular intervals of time given the values of 'u' and 'a'. The program should provide the flexibility to the user to select his own time intervals and repeat the calculations for different values of 'u' and 'a'.

**Description:**

The total distance travelled by vehicle in 't' seconds is given by distance = \( ut + \frac{1}{2}at^2 \) where 'u' and 'a' are the initial velocity (m/sec.) and acceleration (m/sec2).

**Algorithm:**

Step 1: Start

Step 2: Read t , dt

Step 3: Set i to 1

Step 4: Set k to dt

Step 5: Read u,a

Step 6: set s to u*k+0.5*d*k*k

Step 7: Write s

Step 8: If(k<=t) and i=1 then

Begin

Step 8.1 go to step 6

And

Else

Begin
Step 8.2 : read
Step 8.3 : if(j=0) then
Begin
Step 8.3.1: Set I to 0
End
Else

Begin
Step 8.3.2: Set I to 1
Step 8.3.3: go to step 4
End

End

Step 9: Stop

Step 10: End
Flowchart:

Start

Read t, dt

i = 1

k = dt

Read c, a

s = u * k + 1/2 * a * k * k

print a

kk = 1

if k <= i & f
  i = 1

Read j

is
  j = 0

no
  yes

jj = 1

if j = 0
  no

else
  yes

Stop
Program:

```c
#include<stdio.h>
main()
{
    int a,u,t,t1,t2,i;
    float s;
    clrscr();
    printf("ENTER THE VALUES OF a,u,t,t1,t2:");
    scanf("%d%d%d%d%d",&a,&u,&t,&t1,&t2);
    for(i=t1;i<=t2;i=i+t) // performing the looping operation for time intervals
    {
        s=(u*i)+(0.5*a*i*i);  // calculate the total distance
        printf("the distance travelled in %d seconds is %f ",i,s);
    }
    getch();
}
```

Input/Output:

1. ENTER THE VALUES OF a,u,t,t1,t2: 1
   2
   3
   1
   5
   the distance travelled in 1 seconds is 2.500000
   the distance travelled in 4 seconds is 16.000000
2. ENTER THE VALUES OF a,u,t,t1,t2: 0
   1
   2
   3
   4
   the distance travelled in 3 seconds is 3.000000

conclusion: The program is error free

VIVA QUESATIONS:
1) How many types of arrays are there?
   Ans: Three types. They are one dimensional, two dimensional and multi dimensional arrays
**Objective: 13**

Two integer operands and one operator form user, performs the operation and then prints the result.

(Consider the operators +, -, *, /, % and use Switch Statement)

**Description:**
To take the two integer operands and one operator from user to perform the some arithmetic operations by using the following operators like +, -, *, /, %

Ex: 2+3=5

**Algorithm:**

Step 1: Start

Step 2: Read the values of a, b and operator

Step 3: if the operator is ‘+’ then
   R = a + b
   Go to step 8
   Break

Step 4: Else if the operator is ‘-’ then
   R = a - b
   Go to step 8

Step 5: Else if the operator is ‘*’ then
   R = a * b
   Go to step 8

Step 6: Else if the operator is ‘/’ then
   R = a / b
   Go to step 8

Step 7: Else if the operator is ‘%’ then
   R = a % b
   Go to step 8

Step 8: write R

Step 9: End
#include<stdio.h>
main()
{
    char op;
    float a,b,c;
    clrscr();
    printf("enter two operands:");
    scanf("%d%d",&a,&b);
    printf("enter an operator:");
    scanf("%c",&op);
    switch(op) // used to select particular case from the user
    {
        case '+':printf("sum of two numbers %2d %2d is:     %d",a,b,a+b);
            break;
        case '-':printf("subtraction of two numbers %2d %2d is:
                        %d",a,b,a-b);
            break;
        case '*':printf("product of two numbers %2d %2d is:
                        %d",a,b,a*b);
            break;
        case '/':printf("quotient of two numbers %2d %2d is:
                        %d",a,b,a/b);
            break;
        case '%':printf("reminder of two numbers %2d %2d is:
                        %d",a,b,c);
            break;
        default:printf("please enter correct operator");
            break;
    }
    getch();
}

Input/Output:

1. enter two operands: 2 3
   enter an operator: +
   sum of two numbers 2 3 is: 5

2. enter two operands: 3 4
   enter an operator: -
   subtraction of two numbers 3 4 is: -1

3. enter two operands: 3 5
enter an operator:*  
product of two numbers  3  5 is: 15

4. enter two operands: 5 2  
enter an operator:/  
quotient of two numbers  5  2 is: 2
5. enter two operands: 5 2  
enter an operator:%  
reminder of two numbers  5  2 is: 1  

**conclusion:** The program is error free

**VIVA QUESTIONS:**

1) What are the various types of arithmetic operators?  
   Ans: addition (+), multiplication(*), subtraction (-), division(/), modulo(%).

2) What are the types of relational operators?  
   Ans: less than(<), greater than(>), less than or equal to(<=), equal to(==), etc..

3) What are the types of logical operators?  
   Ans: logical AND (&&), logical OR(||), logical NOT(!)
Objective: 14
To find both the largest and smallest number in a list of integers

Description:
This program contains n number of elements, in these elements we can find the largest and smallest numbers and display these two numbers

Algorithm:
Step 1: start
Step 2: read n
Step 3: initialize i=0
Step 4: if i<n do as follows. If not goto step 5
   Read a[i]
   Increment i
   Goto step 4
Step 5: min=a[0], max=a[0]
Step 6: initialize i=0
Step 7: if i<n do as follows. If not goto step 8
   If a[i]<min
   Assign min=a[i]
   Increment i goto Step 7
Step 8: print min,max
Step 9: stop
Flowchart:

1. Start
2. Read n, a[i], min, max,
3. Min = a[0]
   Max = a[0]
4. i = 0
5. i < n
6. i++
7. Read a[i]
8. If A[i] > max
   Max = a[i]
9. If Max < min
   Min = a[i]
10. Print min, max
11. Stop
**Program:**

```c
#include<stdio.h>
void main()
{
    int a[10],i,n,min,max;
    clrscr();
    printf("enter the array size:");
    scanf("%d",&n);
    printf("Enter the elements of array ");
    for(i=0;i<n;i++)  // read the elements of an array
        scanf("%d",a[i]);
    min=a[0];
    max=a[0];
    for(i=0;i<n;i++)// read the elements of an array
    {
        if(a[i]<min)// check the condition for minimum value
            min=a[i];
        if(a[i]>max)//check the condition for maximum value
            max=a[i];
    }
    printf("maximum value is:%d\n",max);
    printf("minimum value is:%d\n",min);
    getch();
}
```

**Output:**

1. enter the array size: 4
   Enter the elements of array 36 13 2 45
   maximum value is: 45
   minimum value is: 2

2. enter the array size: 5
   Enter the elements of array 6 2 1 3 8
   maximum value is: 8
   minimum value is: 1

3. enter the array size: 5
   Enter the elements of array -6 -9 2 5
   maximum value is: 9
   minimum value is: -9

**Conclusion:** the program is error free

**Viva Questions:**

1. What is an array?
   Ans: The collection of similar elements is called array

2. How many types of arrays are there?
   Ans: Three types. They are one dimensional, two dimensional and multi dimensional arrays
**Objective:** To perform the addition of two matrices

**Description:**
program takes the two matrixes of same size and performs the addition an also takes the two matrixes of different sizes and checks for possibility of multiplication and perform multiplication if possible.

**algorithm:**
Step 1: start
Step 2: read the size of matrices A,B – m,n
Step 3: read the elements of matrix A
Step 4: read the elements of matrix B
Step 5: select the choice for you want. If you select case 1 then goto matrix addition. Else goto Step 7.
Step 6: print Sum of matrix A and B
Step 7: if you select case 2 then goto matrix multiplication
Step 8: check if n=p, if not print matrices can not be multiplied
Step 9: Otherwise perform the multiplication of matrices
Step 10: Print the resultant matrix
Step 11: Stop
Flow chart:

start

Declare a[], b[], c[], ch, I, j, k, m, n, p, q, r1, c1

Read the choice ch

If ch <= 2 & ch > 0

Print valid choice

Switch ch

Case 1

Read the size of A, B matrix m, n

I = 0

l < r1

i++

J = 0

j < c1

ji++

Read A[i][j]

I = 0

l < r1

i++

B

Case 2

Read the size of A matrix m, n

I = 0

l < r1

i++

j = 0

j < c1

j++

Read A[i][j]

Read size of matrix B: p, q

I = 0

l < p

i++

B
I=0
I< r1  i++

j=0
j< c1  j++

Print A[i][j]+B[i][j]

Stop

j=0
j< q  j++

Read B[i][j]

If n==p
Matrix cannot be multiplied

i=0
i<m  i++

j=0
j< q  j++

C[i][j]=0

k=0
j< n  k++

C[i][j]=C[i][j]+A[i][k]*B[k][j]

i=0
i<m  i++

j=0
j<q  j++

Print C[i][j]
Program:

#include<stdio.h>

void main()
{
  int ch,i,j,m,n,p,q,k,r1,c1,a[10][10],b[10][10],c[10][10];
  clrscr();
  printf("*****************************************************");
  printf("\n\n\nMENU\n**********************************");
  printf("\n\n\n[1]ADDITION OF TWO MATRICES\n[2]MULTIPLICATION OF TWO MATRICES\n[0]EXIT\n**********************************");
  printf("\n\nEnter your choice:\n");
  scanf("%d",&ch);
  if(ch<=2 & ch>0)
  {
    printf("Valid Choice\n");
  }
  switch(ch)
  {
    case 1:
      printf("Input rows and columns of A & B Matrix:"");
      scanf("%d%d",&r1,&c1);
      printf("Enter elements of matrix A:\n");
      for(i=0;i<r1;i++)
        {
          for(j=0;j<c1;j++)
            scanf("%d",&a[i][j]);
        }
      printf("Enter elements of matrix B:\n");
      for(i=0;i<r1;i++)
        {
          for(j=0;j<c1;j++)
            scanf("%d",&b[i][j]);
        }
      printf("\n =====Matrix Addition=====
");
      for(i=0;i<r1;i++)
        {
          for(j=0;j<c1;j++)
            printf("%5d",a[i][j]+b[i][j]);
        }
  }
printf("\n");
}  
break;

case 2:
printf("Input rows and columns of A matrix:");
scanf("%d%d",&m,&n);
printf("Input rows and columns of B matrix:");
scanf("%d%d",&p,&q);
if(n==p)
{
  printf("matrices can be multiplied\n");
  printf("resultant matrix is %d*%d\n",m,q);
  printf("Input A matrix\n");
  read_matrix(a,m,n);
  printf("Input B matrix\n");
  /*Function call to read the matrix*/
  read_matrix(b,p,q);
  /*function for Multiplication of two matrices*/
  printf("\n =====Matrix Multiplication====\n");
  for(i=0;i<m;++i)
    for(j=0;j<q;++j)
    {
      c[i][j]=0;
      for(k=0;k<n;++k)
        c[i][j]=c[i][j]+a[i][k]*b[k][j];
    }
  printf("Resultant of two matrices:\n");
  write_matrix(c,m,q);
}
/*end if*/
else
{
  printf("Matrices cannot be multiplied.");
}
/*end else*/
break;

case 0:
printf("\n Choice Terminated");
exit();
break;

default:
printf("\n Invalid Choice");
/*Function read matrix*/
int read_matrix(int a[10][10],int m,int n)
{
    int i,j;
    for(i=0;i<m;i++)
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);
    return 0;
}

/*Function to write the matrix*/
int write_matrix(int a[10][10],int m,int n)
{
    int i,j;
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
            printf("%5d",a[i][j]);
        printf("\n");
    }
    return 0;
}

Output:
1. "************************************
   MENU
   ***********************************
   [1]ADDITION OF TWO MATRICES
   [2]MULTIPLICATION OF TWO MATRICES
   [0]EXIT
   ***********************************
   Enter your choice:
   1
Valid Choice
Input rows and columns of A & B Matrix:2
2
Enter elements of matrix A:
2
2
2
Enter elements of matrix B:

2 2 2 2

=====Matrix Addition=====

4 4
4 4

************************************

MENU

**********************************

[1] ADDITION OF TWO MATRICES
[2] MULTIPLICATION OF TWO MATRICES
[0] EXIT

**********************************

Enter your choice:

2
Valid Choice

Input rows and columns of A matrix: 2
3
Input rows and columns of B matrix: 2
2
Matrices cannot be multiplied.

************************************

MENU

**********************************

[1] ADDITION OF TWO MATRICES
[2] MULTIPLICATION OF TWO MATRICES
[0] EXIT

**********************************

Enter your choice:

2
Valid Choice

Input rows and columns of A matrix: 2
2
Input rows and columns of B matrix: 2
2
Matrices can be multiplied
resultant matrix is 2*2

Input A matrix

2
2
2
2
Input B matrix
2
2
2
2

=====Matrix Multiplication=====
Resultant of two matrices:
8  8
8  8
Conclusion: The program is error free

VIVA QUESATIONS:

1) What is condition for performing an matrix addition?
   Ans: program takes the two matrixes of same size and performs the addition
2) What is condition for performing an matrix addition?
   Ans: The two matrixes of different sizes and checks for possibility of multiplication
   and perform multiplication if possible
**Objective : 16**

Functions to insert a sub string into given main string from a given position

**Description:**
In this program we need to insert a string into another string from a specified position.

**Algorithm:**
Step 1: start
Step 2: read main string and sub string
Step 3: find the length of main string(r)
Step 4: find length of sub string(n)
Step 5: copy main string into sub string
Step 6: read the position to insert the sub string( p)
Step 7: copy sub string into main string from position p-1
Step 8: copy temporary string into main string from position p+n-1
Step 9: print the strings
Step 10: stop
Flow chart:

Start

Read the strings A & B

I = 0

I < r

C[i] = A[i]

I ++

S = n + r
O = p + n

I = p                        I++

I < s

X = C[i]

If t < n

A

B
\( A[i] = B[t] \)
\( T = t + 1 \)
\( A[o] = x \)
\( O = o + 1 \)
Print output
Stop
Program:
#include <stdio.h>
#include <conio.h>
#include <string.h>

void main()
{
    char a[10];
    char b[10];
    char c[10];
    int p=0,r=0,i=0;
    int t=0;
    int x,g,s,n,o;
    clrscr();

    puts("Enter First String: ");
    gets(a);
    puts("Enter Second String:");
    gets(b);
    printf("Enter the position where the item has to be inserted: ");
    scanf("%d",&p);
    r = strlen(a);
    n = strlen(b);
    i=0;

    // Copying the input string into another array
    while(i <= r)
    {
        c[i]=a[i];
        i++;
    }
    s = n+r;
    o = p+n;

    // Adding the sub-string
    for(i=p;i<s;i++)
    {
        x = c[i];
        if(t<n)
        {
            a[i] = b[t];
            t=t+1;
        }
        a[o]=x;
        o=o+1;
    }
}
```c
)
  printf("%s", a);
  getch();
}

Output:
1. enter first string:
   computer
2. enter second string:
   gec
3. enter the position where the item has to be inserted: 3
   comgecputer
conclusion: the program is error free

VIVA QUESTIONS:

1) What is string?
   Ans: A string is an collection of characters
2) Which command is used to combined the two strings?
   Ans: Strcat()
3) Which command is used to copy the strings?
   Ans: By using the strcpy() function copies one string to another
Objective: To delete n characters from a given position in a given string

Description: In this program we need to delete a string from the given string at a specified position.

Algorithm:
Step 1: start
Step 2: read string
Step 3: find the length of the string
Step 4: read the value of number of characters to be deleted and positioned
Step 5: string copy part of string from position to end, and (position+number of characters to end)
Step 6: stop
Flow chart:

Start

Read string

Read position, no of characters

Delchar( string, n, pos)

Stop

Subprogram Delchar( )

If ((a+b-1_<=

strlen(x))

Strcpy(&x[b-1],&x[a+b-1])

Puts(x)

Return to mainprogram
Program:
#include <stdio.h>
#include <conio.h>
#include <string.h>

void delchar(char *x, int a, int b);

void main()
{
    char string[10];
    int n, pos, p;
    clrscr();

    puts("Enter the string");
    gets(string);
    printf("Enter the position from where to delete");
    scanf("%d", &pos);
    printf("Enter the number of characters to be deleted");
    scanf("%d", &n);
    delchar(string, n, pos);
    getch();
}

// Function to delete n characters
void delchar(char *x, int a, int b)
{
    if ((a + b - 1) <= strlen(x))
    {
        strcpy(&x[b - 1], &x[a + b - 1]);
        puts(x);
    }
}

Output:
1. Enter the string
   nagaraju
   Enter the position from where to delete: 4
   Enter the number of characters to be deleted: 3
   nagju
2. Enter the string
   kaliraju
   Enter the position from where to delete: 0
   Enter the number of characters to be deleted: 4
   Raju

Conclusion: the program is error free
VIVA QUESTIONS:

1) Which command is used to delete the strings ?
   Ans: delstr();

2) What are the various types of string functions ?
   Ans: Strcat(), strcpy(), delstr(), substr(), strlen() etc..,
Objective: 18
To determine if the given string is a palindrome or not

Description:
if the reverse of a string is equal to original string then it is called palindrome

Algorithm:

Step 1: start
Step 2: read the string
Step 3: store reverse of the given string in a temporary string
Step 4: compare the two strings
Step 5: if both are equal then print palindrome
Step 6: otherwise print not palindrome
Step 7: stop
Flow chart:

Start

Read string

If(Is_palindrome(string))

If(Is_palindrome(string)) true

Print palindrome

Print not palindrome

Stop
Ispalindrome()

Enum Boolean matched = true

If len == 0

Return 0

Left = 0

Right = len - 1

While (left <= right && matched)

If (string[left] != string[right])

Matched = false

Left ++
Right --

Return matched

Return to main program
Program:
#include<stdio.h>
#include<string.h>

enum Boolean {false,true};
enum Boolean IsPalindrome(char string[])
{
    int left,right,len=strlen(string);
    enum Boolean matched=true;
    if(len==0)
        return 0;
    left=0;
    right=len-1;
    while(left<right&&matched)
    {
        if(string[left]!=string[right])
            matched=false;
        else
        {
            left++;
            right--;
        }
    }
    return matched;
}

int main()
{
    char string[40];
    clrscr();
    printf("****Program to test if the given string is a palindrome****\n");
    printf("Enter a string:\n");
    scanf("%s",string);
    if(IsPalindrome(string))
        printf("The given string %s is a palindrome\n",string);
    else
        printf("The given string %s is not a palindrome\n",string);
    getch();
    return 0;
}
Output:
1. Enter the string: malayalam
   The given string malayalam is a palindrome
2. Enter the string: india
   The given string india is not a palindrome
Conclusion: The program is error free

VIVA QUESTIONS:

1) What is meant by palindrome?
   Ans: If the reverse of a string/number is equal to original string/number then it is called palindrome.

2) What is the use of gets() function?
   Ans: To read the string at a time
3) What is the use of puts() function?
   Ans: To write the string at a time
**Objective: 19**

Program that displays the position or index in the string S where the string T begins, or -1 if S doesn’t contain T

**Algorithm:**

Step 1: start
Step 2: read the string and then displayed
Step 3: read the string to be searched and then displayed
Step 4: searching the string T in string S and then perform the following steps
   i. found=strstr(S,T)
   ii. if found print the second string is found in the first string at the position. If not goto step 5
Step 5: print the -1
Step 6: stop
Flow chart:

Start

Initialize s[], t[], found variables

Read first string

Display the string

Read string to be searched

Display the string

Found=strstr(s, t)

If found

no

Print -1

yes

Print the string Found-s

Stop
**Program:**
```c
#include<stdio.h>
#include<string.h>
#include<conio.h>

void main()
{
    char s[30], t[20];
    char *found;
    clrscr();

    /* Entering the main string */
    puts("Enter the first string: ");
    gets(s);

    /* Entering the string whose position or index to be displayed */
    puts("Enter the string to be searched: ");
    gets(t);

    /*Searching string t in string s */
    found=strstr(s,t);
    if(found)
        printf("Second String is found in the First String at %d position.\n",found-s);
    else
        printf("-1");
    getch();
}
```

**Output:**
1. enter the first string:
   kali
   Enter the string to be searched:
   li
   second string is found in the first string at 2 position
2. enter the first string:
   nagaraju
   Enter the string to be searched:
   raju
   second string is found in the first string at 4 position
3. enter the first string:
   nagarjuna
   Enter the string to be searched:
   ma
   -1

**Conclusion:** The program is error free
VIVA QUESTIONS:

1) What is the difference between printf() and puts()?
   Ans: puts() is used to display the string at a time and it doesn’t take any integers values but printf() takes any values as defined by the user.
2) Define pointer variable?
   Ans: Pointer variables are defined as variables that contain the memory addresses of data or executable code.
3) What is use of the strcmp() function?
   Ans: This function compares two strings character by character and returns a value 0 if both strings are equal and non-zero value if the strings are different.
Objective: 20
To count the lines, words & characters in a given text

Description:
In this program we have to count the no of lines, no of words and no of characters in a given program or given text by using the string function

Algorithm:
Step 1: Start
Step 2: Read the text until an empty line
Step 3: Compare each character with newline char ‘\n’ to count no of lines
Step 4: Compare each character with tab char ‘\t’ or space char ‘ ‘ to count no of words
Step 5: Compare first character with NULL char ‘\0’ to find the end of text
Step 6: No of characters = length of each line of text
Step 7: Print no of lines, no of words, no of chars
Step 8: Stop
Initialize end=0, chars=0, words=0, lines=0

While
End==
0
C = 0
If (ctr=getchar())!
="\n"
true
false
Line[c++]=ctr
true
Line[c]=\0
false
If line[0]=\0,
Lines++
Words ++
Chars+=strlen(line)
false
Stop
#include <stdio.h>

main()
{
    char line[81], ctr;
    int i,c,
        end = 0,
        characters = 0,
        words = 0,
        lines = 0;

    printf("KEY IN THE TEXT.\n");
    printf("GIVE ONE SPACE AFTER EACH WORD.\n");
    printf("WHEN COMPLETED, PRESS 'RETURN'.\n\n");

    while( end == 0 )
    {
        /* Reading a line of text */
        c = 0;
        while((ctr=getchar()) != '\n')
        {
            line[c++] = ctr;
            line[c] = '\0';
        }

        /* counting the words in a line */
        if(line[0] == '\0')
        {
            break;
        }
        else
        {
            words++;
            for(i=0; line[i] != '\0';i++)
            {
                if(line[i] == ' ' || line[i] == '	')
                {
                    words++;
                }
            }
        }

        /* counting lines and characters */
        lines = lines +1;
        characters = characters + strlen(line);
    }

    printf("\n");

    printf("Number of lines = %d\n", lines);
    printf("Number of words = %d\n", words);
    printf("Number of characters = %d\n", characters);
}

Output

1. KEY IN THE TEXT.
   GIVE ONE SPACE AFTER EACH WORD.
   WHEN COMPLETED, PRESS 'RETURN'.
Admiration is a very short-lived passion.
Admiration involves a glorious obliquity of vision.
Always we like those who admire us but we do not
like those whom we admire.
Fools admire, but men of sense approve.
Number of lines = 5
Number of words = 36
Number of characters = 205
**Conclusion:** The program is error free

**VIVA QUESATIONS:**

1) What is use of strlen()?
   Ans: to read a string length
2) what is the use of getc() function?
   Ans: To read the character one by one.
3) What is the use of strstr()?
   Ans: The function strstr() searches one string for the occurrence of another. It accepts two strings as parameters and searches the first string for an occurrence of the second
Objective 21:
To generate Pascal’s triangle

Description:
Pascal’s triangle which is used for a coefficient in the equation in polynomials.

Algorithm:
Step 1: Start
Step 2: Initialize m=0
Step 3: Read n
Step 4: If m<n goto step 5, if not goto step 12
Step 5: initialize i=40-m
Step 6: If i>0 is true do as follows. If not goto step 7
   i. print white space
   ii. decrement i
   iii. goto Step 6
Step 7: Initialize j=0
Step 8: If j=m do as follows. If not goto Step 10
   i) if(j==0||m==0)
   ii) Initialize b=1 if not b=b*(m-j+1)/j
   iii) Print white space, b
   iv) Goto Step 9
Step 9: increment j, goto Step 8
Step 10: print new line control
Step 11: increment m, goto step 4
Step 12: Stop
Flow chart:

Start

Read p

I=0 \quad I < p

I++

true

R=40-I
r--

r>0

Print white space

false

X=0 \quad x<=i

X++

true

If x==0

If I==0

true

false

B=1

B=b*(i-x+1)/x

false

true

Print new line

Stop

F

Ptint b
Program:

```c
#include<stdio.h>
#include<conio.h>
void main()
{
    int i,p,r,x,binom=1;
    clrscr();
    printf("enter the how many lines to print");
    scanf("%d", &p);
    i=0;
    while(i<p) // check the condition
    {
        for(r=40-i;r>0;r--)  // perform the looping operation until 0
            printf(" ");
        for(x=0;x<=i;x++)
        {
            if((x==0)||(i==0)) // check the condition
                binom=1;
            else
                binom=binom*(i-x+1)/x;
            printf("%d",binom);
            printf(" ");
        }
        printf("\\n");
        i++;
    }
    getch();
}
```

Output:
1. enter the how many lines to print
   5
   1
   1 1
   1 2 1
   1 3 3 1
   1 4 6 4 1

2. enter the how many lines to print
   3
   1
   1 1
   1 2 1

Conclusion: the program is error free
**VIVA QUESTIONS:**

1) What is meant by Pascal’s triangle?

**Ans:** Pascal’s triangle which is used for a coefficient in the equation in polynomials

2) Define structure?

**Ans:** A structure in C is a heterogeneous user defined data type. A structure may contain different data types. It groups variables into a single entity.
Objective 22:
To construct a pyramid of numbers

Description:
In this program we have to construct output in the pyramid shape manner

Algorithm:
Step 1: Start
Step 2: Initialize the num, l, y, x=35
Step 3: Read the num
Step 4: Perform the loop operation
   For(y=0; y<=num; y++)
Step 5: Gotoxy(x, y+1)
Step 6: Perform the loop operation for displaying digits towards the left and right
   For(i=0-y; i<=y; i++)
Step 7: Print abs(i);
Step 8: x=x-2;
Step 9: Stop
Flow chart:

Start

Initialize num, I, y, x = 35

Read num

Y = 0  y <= num

y ++

true

Gotoxy(x, y + 1)

I = 0  i ++

I <= y

X = x - 2

Print abs(i)

Stop
**Program:**

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

void main()
{
    int num,i,y,x=35;
    clrscr();
    printf("Enter the number to generate the pyramid:\n");
    scanf("%d",&num);

    for(y=0;y<=num;y++)
    {
        /*(x-coordinate,y-coordinate)*/
        gotoxy(x,y+1);

        /*for displaying digits towards the left and right of zero*/
        for(i=0-y;i<=y;i++)
        {
            printf("%3d",abs(i));
            x=x-3;
        }
    }
    getch();
}
```

**Output:**

1. enter the number: 0
   
   4
   
   1 0 1
   
   2 1 0 1 2
   
   3 2 1 0 1 2 3
   
   4 3 2 1 0 1 2 3 4

2. enter the number: 0
   
   3
   
   1 0 1
   
   2 1 0 1 2
   
   3 2 1 0 1 2 3

**Conclusion:** The program is error free

**VIVA QUESTIONS:**

1) What is the use of dot operator in structures?
   Ans: The use of dot(.) operator to access the members of a structure independently. The dot operator connects a member with the structure variable.

2) Define unions?
   Ans: A union is a data type in C which allows the overlay of more than one variable in the same memory area.
Objective 23:
To read in two numbers x and n and then compute the sum of this geometric progression \(1+x+x^2+x^3+\ldots+x^n\)

Description:
In this program we have to read the two numbers and calculate the sum of this geometric progression in above mention.

Algorithm:
Step 1: Start
Step 2: read values of x and n, sum-1, i=1
Step 3: check for n & X
  i) if n<=0 || x<=0
  ii) print values are not valid
  iii) read values of x and n
Step 4: perform the loop operation
  i) for(i=1;i<=n;i++) then follows
  ii) sum=sum+pow(x,i)
Step 5: print sum
Step 6: Stop
Start

Initialize Sum,I,x,n

Read x,n

If n<=0|| x<=0

Print not valid

true

false

I=1                   i<=n

I++

Sum = 1

false

true

Sum=sum+pow(x,i)

Print sum

Stop

Flow chart:
Program:

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

void main()
{
    int s_sum,i,x,n;

    clrscr();
    printf("Enter the values for x and n: ");
    scanf("%d %d", &x, &n);

    if(n<=0 || x<=0)
    {
        printf("Value is not valid\n");
    }
    else
    {
        printf("Value is valid\n");
        s_sum=1;
        for(i=1;i<=n;i++)
        {
            s_sum=s_sum+pow(x,i);
        }
        printf("Sum of series=\%d\n", s_sum);
    }
    getch();
}
```

Output:
1. Enter the values for x and n: 2
   Value is valid
   Sum of series=15
2. Enter the values for x and n: 4
   Value is valid
   Sum of series=21845
3. Enter the values for x and n: 0
   Value is not valid

Conclusion: the program is error free
VIVA QUESATIONS:

1) what are the difference between structures and unions?
Ans: Here the major difference is with in the structure all elements must be allocated memory. But in union highest memory allocation must be allocated the all these elements.
Objective 24:
To convert the given binary number to 2’s complement

Description:
In this program the given binary number is first covert the numbers 0 to 1 and 1 to 0. And finally add the 1 to the converted number. Then we will get the 2’s complement number.

Algorithm:
main program
Step 1: Start
Step 2: declare the subprogram “complement(char *a)”
Step 3: initialize the variable i
Step 4: read the binary number
Step 5: perform the loop operation. if it is true then follows. if not goto step 7
   i) for(i=0;a[i]!='$0';i++)
   ii) if(a[i]!='$0'&&a[i]!='$1') then displayed the number is not valid.
       enter the correct number.
   iii) Exit the loop
Step 6: call sub program ‘complemt(a)’
Step 7: stop

Sub program:
Step 1: initialize the variable I,c=0,b[160
Step 2: l=strlen(a)
Step 3: perform the loop operation. if it is true then follows. if not goto
   i) for(i=l-1;i>=0;i--)
      ii) if(a[i]=='0') then b[i]='1' else
      iii) b[i]='0'
Step 4: for(i=l-1;i>=0;i--) is true
   i) if(i==l-1) then
      ii) if(b[i]=='0') then b[i]='1' else
      iii) b[i]='0',c=1 if not goto step 5
Step 5: if(c==1&&b[i]=='0’) is true then
   i) b[i]='1’, c=0 if not goto Step 6
Step 6: if(c==1&&b[i]=='1’) then b[i]='0’,c=1
Step 7: displayed b[l]="0’
Step 8: print b and return to main program
Flow chart:

Start

Initialize I, a[

Read a

I = 0  i++

A[i]!=\0

If a[i]!=0&&a[i]!=1

Number is not valid

Call sub program Complement(a)

Stop
Complement

Initialize l, c=0, b[

l = strlen(a)

l = l - 1

i = i - 1

If a[i] == '0'

B[i] = '1'

true

false

If i == l - 1

If b[i] == '0'

B[i] = '1'

true

false

C = 1

true

false

If c == 1 && B[i] == '0'

B[i] = '1'

true

false

B[i] = '1'

true

false

C = 0

B
Program:
#include <stdio.h>
#include<conio.h>

void complement (char *a);

void main()
{
    char a[16];
    int i;
    clrscr();
    printf("Enter the binary number");
    gets(a);
    for(i=0;a[i]!="\0"; i++)
    {
        if (a[i]!="0" && a[i]!="1")
        {
            printf("The number entered is not a binary number. Enter the correct number");
            exit(0);
        }
    }
    complement(a);
    getch();
void complement (char *a)
{
    int l, i, c=0;
    char b[16];
    l=strlen(a);
    for (i=l-1; i>=0; i--)
    {
        if (a[i]=='0')
            b[i]='1';
        else
            b[i]='0';
    }
    for(i=l-1; i>=0; i--)
    {
        if(i==l-1)
        {
            if (b[i]=='0')
                b[i]='1';
            else
            {
                b[i]='0';
                c=1;
            }
        }
        else
        {
            if(c==1 && b[i]=='0')
            {
                b[i]='1';
                c=0;
            }
            else if (c==1 && b[i]=='1')
            {
                b[i]='0';
                c=1;
            }
        }
    }
    b[l]='\0';
    printf("The 2's complement is \%s", b);
Output:

1. Enter the binary number 101010
   The 2's complement is 010110

Enter the binary number 11111
The 2's complement is 00001
Enter the binary number 2222
The number entered is not a binary number. Enter the correct number

Conclusion: the program is error free

VIVA QUESTIONS:

1) Expand ASCII?
   Ans: American standard code for information interchange

2) What is binary number?
   Ans: The number which contains only 0 and 1 is called binary number.

3) Define 2’s complement?
   Ans: The given binary number is first covert the numbers 0 to 1 and 1 to 0. And finally add the 1 to the converted number. Then we will get the 2’s complement number.
**Objective 25:**
To convert roman number to it’s decimal equivalent

**Description:**
In this program we have to take the roman value. This value is converted into it’s equivalent decimal number.
Ex: \(X=10\)

**Algorithm:**
Step 1: Start
Step 2: read the roman numerical as string
Step 3: find length of roman numerical
Step 4: for each character in the string
  i) if(char=I) then decimal=1
  ii) if(char=V) then decimal=5
  iii) if(char=X) then decimal=10
  iv) if(char=L) then decimal=50
  v) if(char=C) then decimal=100
  vi) if(char=D) then decimal=500
  vii) if(char=M) then decimal=1000
  viii) otherwise invalid character
Step 5: repeat step 4 until the length of the string
Step 6: k=char[length-1]
Step 7: for each character of decimal string
  i) if(decimal[i]>dec[i-1]) then k=k-decimal[i-1]
  ii) else if(decimal[i]=decimal[i-1] or decimal[i]<decimal[i-1]) then
      k=k+decimal[i-1]
Step 8: repeat step 7 until the length of decimal string
Step 9: print decimal value
Step 10: Stop
Flow chart:

1. **Start**
   - Read roman value
   - Len = strlen(rom)

2. **I = 0, i++**
   - If i < len
     - If rom[i] == 'I'
       - A[i] = 1
     - True
     - False
     - If rom[i] == 'V'
       - A[i] = 5
     - True
     - False
     - If rom[i] == 'X'
       - A[i] = 10
     - True
     - False

3. **K = a[len-1]**
   - I = len-1, i--
   - If I > 0
     - A[i] > a[i-1]
       - True
       - K = K - a[i-1]
     - False
     - If A[i] = a[i-1] && a[i] < a[i-1]
       - True
       - K = k + a[i-1]
     - False

4. **F**
   - A[i] = a[i-1]
   - True
   - A[i] = a[i-1]
   - False

5. **B**

6. **C**
If \( \text{rom}[i] == 'L' \) then \( \text{A}[i] = 50 \)

If \( \text{rom}[i] == 'C' \) then \( \text{A}[i] = 100 \)

If \( \text{rom}[i] == 'D' \) then \( \text{A}[i] = 500 \)

If \( \text{rom}[i] == 'C' \) then \( \text{A}[i] = 1000 \)

Print invalid

Stop

Print k
Program:

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>

void main()
{
    int *a,len,i,j,k;
    char *rom;

    clrscr();

    printf("Enter the Roman Numeral:");
    scanf("%s",rom);

    len=strlen(rom);

    for(i=0;i<len;i++) // loop will continue until I is not greater than length.
    {
        if(rom[i]=='I')
            a[i]=1;
        else if(rom[i]=='V')
            a[i]=5;
        else if(rom[i]=='X')
            a[i]=10;
        else if(rom[i]=='L')
            a[i]=50;
        else if(rom[i]=='C')
            a[i]=100;
        else if(rom[i]=='D')
            a[i]=500;
        else if(rom[i]=='M')
            a[i]=1000;
        else
        { 
            printf("\nInvalid Value");
            getch();
            exit(0);
        }
    }

    k=a[len-1];
    for(i=len-1;i>0;i--) // loop will continue until I less than zero
```
{  
    if(a[i]>a[i-1]) // check the condition  
        k=k-a[i-1];  
    else if(a[i]==a[i-1] || a[i]<a[i-1])  
        k=k+a[i-1];  
}  
printf("nIts Decimal Equivalent is:");  
printf("%d",k);  
getch();
}  

**Output:**

Enter the Roman Numeral:D

Its Decimal Equivalent is:500
Enter the Roman Numeral:X

Its Decimal Equivalent is:10
Enter the Roman Numeral:23

Invalid Value

**Conclusion:** The program is error free

**VIVA QUESTIONS:**

1) What is difference between structure and unions?
   Ans: The amount of memory required to store a structure variable is the sum of size all
   the members in addition to the padding bytes that may be provided by the compiler. In
   case of a union the amount of memory required is the same as that required by its largest
   member.

2) What are various operations performed on union?
   Ans: i) An union variable can be assigned to another union variable
       ii) A union variable can be passed to a function as a parameter
           iii) The address of the union variable can be extracted by using the address of
               operator (&).
**Objective 26:**
To read the two complex numbers and perform the addition and multiplication of these two numbers.

**Description:**
In this program the complex number means it contains the two parts. first one is real part and second one is imaginary part(2+3i). by taking these two complex numbers we can perform the addition and multiplication operation.

**Algorithm:**
Step 1: Start
Step 2: declare structure for complex numbers
Step 3: read the complex number
Step 4: read choice
Step 5: if choice=1 then addition operation will perform and it contains following steps
   i)  w.realpart = w1.realpart+w2.realpart;
   ii) w.imgpart = w1.imgpart+w2.imgpart; goto step 4
Step 6: if choice=2 then multiplication operation will perform and it contains following steps
   i)  w.realpart=(w1.realpart*w2.realpart)-(w1.imgpart*w2.imgpart);
   ii) w.imgpart=(w1.realpart*w2.imgpart)+(w1.imgpart*w2.realpart); goto step 4
Step 7: if choice=0 then exit operation will perform

Step 8: if w.imgpart>0 then print realpart+imgpart else Print realpart.
Step 9: Stop
Flow chart:

1. Start
2. Declare structure
3. Read option
4. If option=(add or mul)
   a. Read real part of first number
   b. Read image part of first number
   c. Read real part of second number
   d. Read image part of second number
5. If option=ad
   a. w.realpart = w1.realpart + w2.realpart
   b. w.imgpart = w1.imgpart + w2.imgpart
   c. If w.imgpart > 0
      i. true: Print w.realpart + w.imgpart i
      ii. false: Print real part
6. If option=mul
   a. w.realpart = (w1.realpart*w2.realpart)
   b. w.imgpart = (w1.imgpart*w2.imgpart)
   c. If w.imgpart > 0
      i. true: Print w.realpart + w.imgpart i
      ii. false: Print real part
7. Stop
Program:

```
#include<stdio.h>
#include<math.h>
void arithmetic(int opern);
struct comp
{
    double realpart;
    double imgpart;
};
void main()
{
    int opern;
    clrscr();
    printf("\n\t\n\t***** MAIN MENU *****\n\nSelect your option: \n 1 : ADD\n 2 : MULTIPLY\n 0 : EXIT \n\n\t	Enter your Option [  ]\n\n"%d",&opern);
    if(opern>2)
    {
        printf("invalid option");
    }
    else
    {
        switch(opern)
        {
            case 0:
                exit(0);
            case 1:
            case 2:
                arithmetic(opern);
            default:
                main();
        }
    }
    getch();
}
void arithmetic(int opern)
{
    struct comp w1, w2, w;
    printf("\n Enter two Complex Numbers  (x+iy):
 Real Part of First Number:");
    scanf("%f",&w1.realpart);
    printf(" Imaginary Part of First Number:");
    scanf("%f",&w1.imgpart);
    printf(" Real Part of Second Number:");
    scanf("%f",&w2.realpart);
    ```
printf("n Imaginary Part of Second Number:");
scanf("%lf", &w2.imgpart);
switch(opern)
{
    /*addition of complex number*/
    case 1:
        w.realpart = w1.realpart + w2.realpart;
        w.imgpart = w1.imgpart + w2.imgpart;
        break;
    /*multiplication of complex number*/
    case 2:
        w.realpart = (w1.realpart * w2.realpart) - (w1.imgpart * w2.imgpart);
        w.imgpart = (w1.realpart * w2.imgpart) + (w1.imgpart * w2.realpart);
        break;
}
if (w.imgpart > 0)
    printf("n Answer = %lf+%lf\n", w.realpart, w.imgpart);
else
    printf("n Answer = %lf%lf\n", w.realpart, w.imgpart);
getch();
main();

Output:

***** MAIN MENU *****

Select your option:
1 : ADD
2 : MULTIPLY
0 : EXIT

    Enter your Option [ 1 ]

Enter two Complex Numbers (x+iy):
Real Part of First Number:2

Imaginary Part of First Number:2

Real Part of Second Number:2

Imaginary Part of Second Number:2

Answer = 4.000000+4.000000i

***** MAIN MENU *****
Select your option:
1 : ADD
2 : MULTIPLY
0 : EXIT

   Enter your Option [ 2]

Enter two Complex Numbers (x+iy):
Real Part of First Number:2
Imaginary Part of First Number:2
Real Part of Second Number:2
Imaginary Part of Second Number:2

Answer = 0.000000+8.000000i

***** MAIN MENU *****

Select your option:
1 : ADD
2 : MULTIPLY
0 : EXIT

   Enter your Option [ 3]
invalid option

***** MAIN MENU *****

Select your option:
1 : ADD
2 : MULTIPLY
0 : EXIT

   Enter your Option [ 0]

**Conclusion:** The program is error free

**VIVA QUESTIONS:**

1) Define structure?
   Ans: Structure is a method for packing data of different types. Structure help to organize complex data in a more meaningful way.

2) What is use of <math.h> header file?
   Ans: It is used to access the mathematical functions in programs.
**Objective 27:**
Program which copies one file to another

**Description:**
In this program we have to use the file functions to perform the copy operation from one file to another file.

**Algorithm:**
Step 1: Start
Step 2: read command line arguments
Step 3: check if no of arguments =3 or not. If not print invalid no of arguments
Step 4: open source file in read mode
Step 5: if NULL pointer, then print source file can not be open
Step 6: open destination file in write mode
Step 7: if NULL pointer, then print destination file can not be open
Step 8: read a character from source file and write to destination file until EOF
Step 9: Close source file and destination file
Step 10: Stop
Read arguments from command line

If arg! = 3

Print invalid no of arguments

Fs=fopen(arg[1],"r")

If fs=NUL

Print source file can not be opened

Ft=fopen(arg[2],"w")

If ft=NUL

Print target file can not be opened

Fclose(fs)

Ch=fgetc(fs)

If ch==EO

Fclose(fs) Fclose(ft)

If ch==EO

Fputc(ch,ft)

Stop
Program:

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

void main(int argc, char *argv[])
{
    FILE *fs,*ft;
    char ch;
    clrscr();
    if(argc!=3)
    {
        puts("Invalid number of arguments.");
        exit(0);
    }
    fs = fopen(argv[1],"r");
    if(fs==NULL)
    {
        puts("Source file cannot be opened.");
        exit(0);
    }
    ft = fopen(argv[2],"w");
    if (ft==NULL) // check the condition if the file pointer is NULL or not
    {
        puts("Target file cannot be opened.");
        fclose(fs);
        exit(0);
    }
    while(1)
    {
        ch=fgetc(fs);
        if (ch==EOF) // check the condition if the file is end or not
            break;
        else
            fputc(ch,ft);
    }
    fclose(fs);
    fclose(ft);
    getch();
}
**Output:**
source.c
	this is source text
output.c

Command line arguments
source.c output.c
source.c
	this is source text
output.c
	this is source text

Command line arguments
source.c
Invalid number of arguments.

**Conclusion:** the program is error free

**VIVA QUESTIONS:**

1) What is file?
Ans: The collection of alphabets is called file

2) What are the various operations performed on the file?
Ans: fopen(), fread(), fwrite(), fclose() etc.,

3) What is the use of file pointer?
Ans: The file pointer must be used in subsequent operations on the file
**Objective 28:**
To reverse the first n characters in a file

**Description:**
This program perform the reverse operation of n characters in the file

**Algorithm:**
Step 1: Start
Step 2: read the command line arguments
Step 3: check if arguments=3 or not
   If not print invalid no of arguments
Step 4: open source file in read mode
Step 5: if NULL pointer, then print file can not be open
Step 6: Store no of chars to reverse in k
   K= *argv[2]-48
Step 7: read the item from file stream using fread
Step 8: Store chars from last position to initial position in another string(temp)
Step 9: print the temp string
Step 10: Stop
Start

Read command line args

If argc != 3

false

Fp = fopen(argv[1], "r")

true

If fp == NULL

Print file cannot be opened

false

k = *argv[2] - 48
n = fread(a, 1, k, fp)
a[n] = '\0'
len = strlen(a)

true

I = len - 1
i--
I >= 0

S[j+1] = '\0'

false

Stop

Print invalid no of args

Flow chart:
Program:

```c
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <process.h>

void main(int argc, char *argv[])
{
    char a[15];
    char s[20];
    char n;
    int k;
    int j=0;
    int i;
    int len;
    FILE *fp;

    if(argc!=3)
    {
        puts("Improper number of arguments.");
        exit(0);
    }
    fp = fopen(argv[1],"r");
    if(fp == NULL)
    {
        puts("File cannot be opened.");
        exit(0);
    }

    k=*argv[2]-48;
    n = fread(a,1,k,fp);
    a[n]=\0;
    len=strlen(a);
    for(i=len-1;i>=0;i--)
    {
        s[j]=a[i];
        printf("%c",s[j]);
        j=j+1;
    }
    s[j+1]=\0;
    getch();
}
```
**Output:**
source.c
    this is source
output.c

Command line arguments
source.c output.c
source.c
    this is source
ecruos si siht

Command line arguments
source.c
Invalid number of arguments.

**Conclusion:** the program is error free

**VIVA QUESTIONS:**
1) List out the file handling functions?
   Ans: fopen(), fprintf(), fclose(), fscanf(), fgetc(), fputc(), etc..

2) What is the use of fseek() function?
   Ans: The function fseek sets the file pointer associated with a stream to a new position

3) What is use of the fflush() function?
   Ans: If the given stream has a buffered output, fflush writes the output of the stream to the associate file.
Objective 29:
Program that uses functions to perform the following operations on single linked lists. i) creation ii) insertion iii) deletion iv) traversal

Description:
In this program we have to create a single linked list, insert the elements into that list, delete some elements from that list and then perform the sorting operation and traversal operation on that created linked list.

Algorithm:

Step 1: Start

Step 2: Declare a structure named linked-list

Step 3: Declare the pointers next, first, fresh, ptr

Step 4: Print main menu

Step 5: Read choice

Step 6: Switch(choice)

Step 7: If(choice==1)
   7.1 Assign fresh=malloc(size of (node))
   7.2 Read the element fresh->data
   7.3 Read the choice where to insert
   7.4: Switch(choice)
      7.4.1: If choice==1
      7.4.2: Call the function IBegin()
      7.4.3: If choice==2
      7.4.4: Call the function Iend()
      7.4.5: If choice==3
      7.4.6: Call the function Imiddle()

Step 8: If(choice==2)
   8.1: Read the position to delete
   8.2: Switch(choice)
      8.2.1: If choice==1
      8.2.2: Call the function DBegin()
      8.2.3: If choice==2
      8.2.4: Call the function Dend()
      8.2.5: If choice==3
      8.2.6: Call the function Dmiddle()

Step 9: If choice==3
   9.1 Call function view
Step 10: If choice==4
  10.1 Exit()

Step 11: Start insert function

Step 12: If(first==null)

Step 13: First->data=e

Step 14: First->next=null

Step 15: Else declare new node

Step 16: fresh->data=e

Step 17: If choice=1

Step 18: fresh->next=first

Step 19: first=fresh

Step 20: if choice=2

Step 21: ptr=first

Step 22: ptr->next=fresh

Step 23: fresh->next=full

Step 24: If choice =3

Step 25: Enter the position

Step 26: at p-1 node

Step 27: fresh->next= ptr->next

Step 28: ptr->next=fresh

Step 29: for delete function

Step 30: If first!=null

Step 31: Enter the position to delete

Step 32: If choice=1
Step 33: \( d = \text{first} \rightarrow \text{data} \)

Step 34: \( \text{first} = \text{first} \rightarrow \text{next} \)

Step 35: if choice=2

Step 36: \( \text{ptr} = \text{first} \)

Step 37: Traverse to last node

Step 38: \( d = \text{ptr} \rightarrow \text{next} \rightarrow \text{data} \)

Step 39: \( \text{ptr} \rightarrow \text{next} = \text{ptr} \rightarrow \text{next} \rightarrow \text{next} \)

Step 40: Print d value

Step 41: for function view

Step 42: for \( \text{ptr} = \text{first} \) and \( \text{ptr} \neq \text{null} \) and \( \text{ptr} = \text{ptr} \rightarrow \text{next} \)

Step 43: Print \( \text{ptr} \rightarrow \text{data} \)

Step 44: End
Flowchart:

Insert 1

Insert 2

Insert 3 at pos 0

Insert 4 at pos 2

view

Delete (4)

Delete (2)

Delete (3)
Program:

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

int ch,i,n,j,p,item;  /* VARIABLE DECLARATION */

/* START OF STRUCTURE DEFINITION */

struct link
{
    int data;
    struct link *next;
}*start,*new,*l,*l1,*start1,*t;

/* END OF STRUCTURE DEFINITION */

/* START OF MAIN FUNCTION */

main()
{
    clrscr();
    start=NULL;
    start1=NULL;
    printf(" **** MENU **** ");
    printf("\n 1.Insertion
 2.Deletion
 3.Traverse
 4.Search
 5.Sort
 6.Merge
 7.Reverse\n");
    while(1)
    {
        printf("enter the choice:");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1: insert();
            break;
        case 2: delete();
            break;
        case 3: traverse();
            break;
        case 4: search();
            break;
        case 5: sort();
            break;
        case 6: merge();
            break;
        case 7: reverse();
        }
break;
case 8: exit();
}
}
getch();

/* END OF MAIN FUNCTION */

/* START OF INSERT FUNCTION */

insert()
{
l=start;
printf("enter the item to be inserted:");
scanf("%d", &item);
new=malloc(sizeof(struct link));
new->data=item;
if(start==NULL)
{
    new->next=NULL;
    start=new;
}
else
{
    printf("1.start
2.middle
3.end
");
    printf("enter the place to place the item: ");
    scanf("%d", &ch);
    if(ch==1)
    {
        new->next=start;
        start=new;
    }
    if(ch==2)
    {
        printf("enter the position to place item:");
        scanf("%d", &p);
        for(i=1; i<p-1; i++)
            l=l->next;
        new->next=l->next;
        l->next=new;
    }
    if(ch==3)
    {
        while(l->next!=NULL)
            l=l->next;
    }
}
}
new->next=NULL;
l->next=new;
}
}
}

/* END OF INSERT FUNCTION */

/* START OF DISPLAY FUNCTION */

traverse()
{
    if(start==NULL)
        printf("LIST IS EMPTY\n");
    else
    {
        for(l=start;l->next!=NULL;l=l->next)
            if(l==start)
                printf("\nstart:%d->",l->data);
            else
                printf("%7d->",l->data);
        if(l->next==NULL)
            printf(" last:%d->\n",l->data);
    }
}

/* END OF DISPLAY FUNCTION */

/* START OF DELETE FUNCTION */

delete()
{
    l=start;
    if(start==NULL)
        printf("NO ITEMS IN THE LIST\n");
    else
    {
        printf("1.start\n2.middle\n3.end\n");
        printf("enter the place to delete the item: ");
        scanf("%d",&ch);
        if(ch==1)
        {
            item=start->data;
            printf("deleted item is:%d\n",item);
            start=start->next;
        }
        if(ch==2)
printf("enter the position to delete item:");
scanf("%d",&p);
if(l->next==NULL)
{
    item=l->data;
    printf("deleted item is:%d\n",item);
    l=start=NULL;
}
else
{
    for(i=1;i<p-1;i++)
    l=l->next;
    item=l->next->data;
    printf("deleted item is:%d\n",item);
    l->next=l->next->next;
}
}
if(ch==3)
{
    if(l->next==NULL)
    {
        item=l->data;
        printf("deleted item is:%d\n",item);
        l=start=NULL;
    }
    else
    {
        while(l->next->next!=NULL)
        l=l->next;
        item=l->next->data;
        printf("deleted item is:%d\n",item);
        l->next=NULL;
        l=l->next;
    }
}

/* END OF DELETE FUNCTION */
/* START OF SEARCH FUNCTION */

search()
{
    int f=0;

printf("enter the search item:");
scanf("%d",&item);
if(start==NULL)
  printf("LIST IS EMPTY");
else
  {
    for(l=start,i=1;l!=NULL;l=l->next,i++)
      if(l->data==item)
      {
        f=1;
        break;
      }
    if(f==1)
      printf("item %d found at position :%d\n",item,i);
    else
      printf("item %d not found\n",item);
  }
/* END OF SEARCH FUNCTION */
/* START OF SORT FUNCTION */

sort()
{
  int t;
  if(start==NULL)
    printf("LIST IS EMPTY");
  else
    {
      for(l1=start;l1->next!=NULL;l1=l1->next)
      {
        for(l=start;l->next!=NULL;l=l->next)
          if(l->data > l->next->data)
          {
            t=l->data;
            l->data=l->next->data;
            l->next->data=t;
          }
      }
      printf("THE SORTED ORDER IS:");
      for(l=start;l!=NULL;l=l->next)
        printf("%3d",l->data);
      printf("\n");
    }
}
merge()
{
    printf("enter no of elements to be inserted in second list :\n");
    scanf("%d",&n);
    for(j=1;j<=n;j++)
    {
        l1=start1;
        printf("enter the item to be inserted:\n");
        scanf("%d",&item);
        new=malloc(sizeof(struct link));
        new->data=item;
        new->next=NULL;
        if(start1==NULL)
            start1=new;
        else
        {
            printf("1.start\n2.middle\n3.end\n");
            printf("enter the place to place the item:\n");
            scanf("%d",&ch);
            if(ch==1)
            {
                new->next=start1;
                start1=new;
            }
            if(ch==2)
            {
                printf("enter the position to place item:\n");
                scanf("%d",&p);
                for(i=1;i<p-1;i++)
                    l1=l1->next;
                new->next=l1->next;
                l1->next=new;
            }
            if(ch==3)
            {
                while(l1->next!=NULL)
                    l1=l1->next;
                l1->next=new;
            }
        }
    }
}
if(start==NULL)
    start=start1;
else
{
    l=start;
    while(l->next!=NULL)
    l=l->next;
for(l1=start1;l1->next!=NULL;l1=l1->next)
{
    l->next=l1;
    l=l->next;
}
printf(" *** LIST IS MERGED *** \n");
}

/* END OF MERGE FUNCTION */

/* START OF REVERSE FUNCTION */

reverse()
{
    if(start==NULL)
        printf("LIST IS EMPTY\n");
    else
{
        l=start;
        l1=t=NULL;
        while(l!=NULL)
        {
            l1=t;
            t=l;
            l=l->next;
            t->next=l1;
        }
        start=t;
        printf(" *** LIST IS REVERSED *** \n");
    }
}

/* END OF REVERSE FUNCTION */

***** OUTPUT *****
**** MENU ****
1. Insertion
2. Deletion
3. Traverse
4. Search
5. Sort
6. Merge
7. Reverse

enter the choice: 1
enter the item to be inserted: 1
enter the choice: 1
enter the item to be inserted: 2
1. start
2. middle
3. end
enter the place to place the item: 1
enter the choice: 1
enter the item to be inserted: 3
1. start
2. middle
3. end
enter the place to place the item: 3
enter the choice: 1
enter the item to be inserted: 4
1. start
2. middle
3. end
enter the place to place the item: 2
enter the position to place item: 3
enter the choice: 3

start: 2->
  1->
  4->
last: 3->
enter the choice: 4
enter the search item: 4
item 4 found at position: 3
enter the choice: 6
enter no of elements to be inserted in second list: 3
enter the item to be inserted: 5
enter the item to be inserted: 6
1. start
2. middle
3. end
enter the place to place the item: 1
enter the item to be inserted: 7
1. start
2. middle
3. end
enter the place to place the item: 2
enter the position to place item: 2
*** LIST IS MERGED ***
enter the choice: 3

start: 2->
    1->
    4->
    3->
    6->
    7->
last: 5->
enter the choice: 7
*** LIST IS REVERSED ***
enter the choice: 3

start: 5->
    7->
    6->
    3->
    4->
    1->
last: 2->
enter the choice: 4
enter the search item: 1
item 1 found at position: 6
enter the choice: 5
THE SORTED ORDER IS:  1  2  3  4  5  6  7
enter the choice: 2
1. start
2. middle
3. end
enter the place to delete the item: 1
deleted item is: 1
enter the choice: 2
1. start
2. middle
3. end
enter the place to delete the item: 3
deleted item is: 7
enter the choice: 2
1. start
2. middle
3. end
enter the place to delete the item: 2
enter the position to delete item: 4
deleted item is: 5
enter the choice: 3

start: 2->
   3->
   4->
last: 6->
enter the choice: 2
1. start
2. middle
3. end
enter the place to delete the item: 1
deleted item is: 2
enter the choice: 2
1. start
2. middle
3. end
enter the place to delete the item: 2
enter the position to delete item: 2
deleted item is: 4
enter the choice: 3
start: 3->
   last: 6->
enter the choice: 2
1. start
2. middle
3. end
enter the place to delete the item: 2
enter the position to delete item: 2
deleted item is: 6
enter the choice: 2
1. start
2. middle
3. end
enter the place to delete the item: 1
deleted item is: 3
enter the choice: 3
LIST IS EMPTY
enter the choice: 2
NO ITEMS IN THE LIST
enter the choice: 8

conclusion: the program is error free

VIVA QUESTIONS:
1) List out the memory allocation functions?
Ans: malloc(), calloc(), free(), realloc() etc.,

2) Define linked list?
Ans: Linked list is list whose order is given by links from one item to the next

3) List out the advantages of linked list?
**Ans:**
   i) Dynamic data structure
   ii) No waste memory space
   iii) Flexibility
Objective 30:
Program that uses functions to perform the following operations on doubly linked list i) creation ii) insertion iii) deletion iv) traversal

Description: In this program we have to create a doubly linked list, insert the elements in to a doubly linked list, delete the elements from that list and finally perform the traversal operation

Algorithm:

Step 1: Start

Step 2: Declare a structure with *next, *pre

Step 3: Declare *start, *new , *l as structure pointers

Step 4: Print main menu

Step 5: Read choice

Step 6: Switch choice
  6.1: call insert function if choice==1
  6.2: call delete function if choice==2
  6.3: call view function if choice==3

Step 7: Stop

Step 8: Start of insert function

Step 9: Read e

Step 10: If start==null

Step 11: Create a new node

Step 12: Start->data=e

Step 13: Start->next=null

Step 14: Start->pre=null

Step 15: read choice, where to insert

Step 16: if choice==1
  16.1: Create a new mode
  16.2: new -> data=e
Step 16.3: new -> next=start

Step 16.4: start->pre=new

Step 16.5: new->pre=null

Step 16.6: Start->new

Step 17: otherwise if choice==2
17.1: read position p
17.2: l=start
17.3: while i<(p-1)
17.4: increment i
17.5: l=l->next
17.6: new -> data =e
17.7: new -> pre=l
17.8: new->next=new
17.9: l-> next=new
17.10: l->next->pre=new

Step 18: if choice==3
18.1: l=start
18.2: while l->next!=null
18.3: l=l->next
18.4: create a new mode
18.5: new->data=e
18.6: new->next=null
18.7: l->next=new
18.8: new->pre=l

Step19: end of insert function

Step20: start of deletion function

Step21: write menu

Step22: read choice

Step23: if choice==1

23.1: temp=start->data
23.2: start=start->next
23.3: start->pre=null

Step24: if choice==2

24.1: read position
24.2: l=start
24.3: while (i=1 <p-1)
24.4: l=l->next
24.5: increment l by 1
24.6: temp=l-next->data
24.7: l->next=l->next->next
24.8: l->next->pre=l

Step25: if choice==3

25.1: read l=start
25.2: while l->next->next!= null
25.3: l=l->next
25.4: temp=l->next->data
25.5: l->next=null

Step26: end of delete function

Step27: start of view function

Step28: read choice

Step29: if choice==1

29.1: l=next

29.2: while (l->next!= null)

29.3: write l-> data, l=l->next

29.4: write l->data

Step30: if choice==2

30.1: l=start

30.2: while l!=start

30.3: write l->data

30.4: l=l->pre

30.5: write l->data

Step31: end of function view
Program:
#include<stdio.h>
#include<malloc.h>

/* START OF STRUCTURE DEFINITION */

struct link
{
    int data;
    struct link *next;
    struct link *prev;
}*start,*new,*temp,*l,*l1,*t,*start1;

/* END OF STRUCTURE DEFINITION */

int item,ch,i,j,p,n;         /* VARIABLE DECLARATION */

/* START OF MAIN FUNCTION */

main()
{
    start=NULL;
    start1=NULL;
    clrscr();
    printf(" **** MENU ****");
    printf("n1. Insertion
2. Deletion
3. Traverse
4. search
5. sort
6. merge
7. reverse
8. exit
");
    while(1)
    {
        printf("enter your choice:");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1:insert();
            break;
        case 2:delete();
            break;
        case 3:display();
            break;
        case 4:search();
            break;
        case 5:sort();
            break;
        case 6:merge();
            break;
        case 7:reverse();
            break;
        case 8:exit();
            break;
        default:
            printf("invalid choice!");
            break;
        }
    }
}
case 6: merge();
        break;
    case 7: reverse();
        break;
    case 8: exit();
};
}
getch();
}

/ * END OF MAIN FUNCTION * /

/ * START OF INSERT FUNCTION * /

insert()
{
    l=start;
    printf("enter an item to be inserted:");
    scanf("%d",&item);
    new=malloc(sizeof(struct link));
    new->data=item;
    if(start==NULL)
    {
        new->prev=NULL;
        new->next=NULL;
        start=new;
    }
    else
    {
        printf("1.start
2.middle
3.end\n");
        printf("enter the place to insert item:");
        scanf("%d",&ch);
        if(ch==1)
        {
            new->next=start;
            new->prev=NULL;
            start=new;
        }
        if(ch==2)
        {
            printf("enter the position to place item:");
            scanf("%d",&p);
            for(i=1;i<p-1;i++)
                l=l->next;
            new->prev=l;
        }
    }
}
new->next=l->next;
l->next=new;
}
if(ch==3)
{
    while(l->next!=NULL)
        l=l->next;
    new->prev=l;
    new->next=NULL;
    l->next=new;
}
}

/* END OF INSERT FUNCTION */

/* START OF DELETE FUNCTION */
delete()
{
    l=start;
    if(start==NULL)
        printf("*** LIST IS EMPTY ***");
    else
        {
            printf("1.start\n2.middle\n3.end\n");
            printf("enter the place to delete the item:");
            scanf("%d",&ch);
            if(ch==1)
                {
                    item=start->data;
                    printf("deleted item is :%d",item);
                    start=start->next;
                    start->prev=NULL;
                }
            if(ch==2)
                {
                    printf("enter the position to delete an item: ");
                    scanf("%d",&p);
                    if(l->next==NULL)
                        {
                            item=l->data;
                            printf("deleted item is:%d",item);
                            l=start=NULL;
                        }
                    else
                        {
                            item=l->data;
                            printf("deleted item is:%d",item);
                            l=start=NULL;
                        }
                }
            else
{  
    for(i=1;i<p-1;i++)
    l=l->next;
    item=l->next->data;
    printf("deleted item is:%d",item);
    l->next=l->next->next;
    l->next->prev=l;
}
}
if(ch==3)
{
    if(l->next==NULL)
    {
        item=l->data;
        printf("deleted item is :%d",item);
        l->prev=NULL;
        l=start=NULL;
    }
    else
    {
        while(l->next->next!=NULL)
        l=l->next;
        item=l->next->data;
        printf("deleted item is:%d",item);
        l->next=NULL;
    }
}
}

/* END OF DELETE FUNCTION */

/* START OF DISPLAY FUNCTION */
display()
{
    if(start==NULL)
    printf("*** LIST IS EMPTY ***\n");
    else
    {
        for(l=start;l->next!=NULL;l=l->next)
        if(l==start)
            printf("\nstart:%d",l->data);
        else
            printf("\n %8d",l->data);
        if(l->next==NULL)
            printf("\n last:%d",l->data);
    }
}
/* END OF DISPLAY FUNCTION */
/* START OF SEARCH FUNCTION */
search()
{
    int f=0;
    if(start==NULL)
        printf(" *** LIST IS EMPTY *** ");
    else
    {
        printf("enter the search item:");
        scanf("%d",&item);
        for(l=start,i=1;l!=NULL;l=l->next,i++)
            if(item==l->data)
            {
                f=1;
                break;
            }
        if(f==1)
            printf("item %d found at position %d",item,i);
        else
            printf("item %d not found in list",item);
    }
}
/* END OF SEARCH FUNCTION */
/* START OF SORT FUNCTION */
sort()
{
    int t;
    if(start==NULL)
        printf(" *** LIST IS EMPTY *** ");
    else
    {
        for(l1=start;l1->next!=NULL;l1=l1->next)
            for(l=start;l->next!=NULL;l=l->next)
                if(l->data > l->next->data)
                {
                    t=l->next->data;
                    l->next->data=l->data;
                    l->data=t;
                }
        for(l=start;l->next!=NULL;l1=l->next)
            for(l=start;l->next!=NULL;l=l->next)
                if(l->data > l->next->data)
                {
                    t=l->next->data;
                    l->next->data=l->data;
                    l->data=t;
                }
printf("THE SORTED ORDER IS:");
for(l=start;l!=NULL;l=l->next)
printf("%3d",l->data);

printf("\n");

/* END OF SORT FUNCTION */

/* START OF MERGE FUNCTION */

merge()
{
    printf("enter number items to be inserted in second list:");
    scanf("%d",&n);
    for(j=1;j<=n;j++)
    {
        l1=start1;
        printf("enter an item:");
        scanf("%d",&item);
        new=malloc(sizeof(struct link));
        new->data=item;
        if(start1==NULL)
        {
            new->prev=NULL;
            new->next=NULL;
            start1=new;
        }
        else
        {
            printf("1.start\n2.middle\n3.end\n");
            printf("enter the place to insert item:");
            scanf("%d",&ch);
            if(ch==1)
            {
                new->next=start1;
                new->prev=NULL;
                start1=new;
            }
            if(ch==2)
            {
                printf("enter the position to place item:");
                scanf("%d",&p);
                for(i=1;i<p-1;i++)
            }
        }
    }
}

l1=l1->next;
new->prev=l1;
new->next=l1->next;
l1->next=new;
}
if(ch==3)
{
    while(l1->next!=NULL)
    l1=l1->next;
new->prev=l1;
new->next=NULL;
l1->next=new;
}
}
if(start==NULL)
start=start1;
else
{
l=start;
while(l->next!=NULL)
l=l->next;
for(l1=start1;l1->next!=NULL;l1=l1->next)
{
l->next=l1;
l=l->next;
}
printf(" *** LIST IS MERGED *** \n");
}

/* END OF MERGE FUNCTION */

/* START OF REVERSE FUNCTION */

reverse()
{
    if(start==NULL)
    printf(" *** LIST IS EMPTY *** \n ");
else
{
l=start;
l1=t=NULL;
while(l!=NULL)
{
l1=t;
l=l->next;
}
t=l;
l=l->next;
t->next=l1;
}
start=t;
printf(" *** LIST IS REVERSED *** \n");
}

/* END OF REVERSE FUNCTION */

** Input/Output:**

**** MENU ****
1.Insertion
2.Deletion
3.Traverse
4.search
5.sort
6.merge
7.reverse
8.exit
enter your choice: 1 
enter an item to be inserted: 10 
enter your choice: 1 
enter an item to be inserted: 20 
1.start 
2.middle 
3.end 
enter the place to insert item: 1 
enter your choice: 1 
enter an item to be inserted: 30 
1.start 
2.middle 
3.end 
enter the place to insert item: 3 
enter your choice: 1 
enter an item to be inserted: 40 
1.start 
2.middle 
3.end
enter the place to insert item: 2
enter the position to place item: 3
enter your choice: 1
enter an item to be inserted: 50
1. start
2. middle
3. end
enter the place to insert item: 2
enter the position to place item: 2
enter your choice: 3

start: 20
  50
  10
  40
last: 30
enter your choice: 6
enter number items to be inserted in second list: 3
enter an item: 60
enter an item: 70
1. start
2. middle
3. end
enter the place to insert item: 3
enter an item: 80
1. start
2. middle
3. end
enter the place to insert item: 1
*** LIST IS MERGED ***
enter your choice: 3
start: 20
  50
  10
  40
  30
  80
  60
last: 70
enter your choice: 4
enter the search item: 80
item 80 found at position 6
enter your choice:4
enter the search item:10
item 10 found at position 3
enter your choice:7
*** LIST IS REVERSED ***
enter your choice:3
start:70
  60
  80
  30
  40
  10
  50
last: 20
enter your choice:5
THE SORTED ORDER IS: 10 20 30 40 50 60 70 80
enter your choice:2
1.start
2.middle
3.end
enter the place to delete the item:1
deleted item is :10
enter your choice:2
1.start
2.middle
3.end
enter the place to delete the item:3
deleted item is:80
enter your choice:2
1.start
2.middle
3.end
enter the place to delete the item:2
enter the position to delete an item:3
deleted item is:40
enter your choice:3
start:20
  30
  50
  60
last: 70
enter your choice:2
1.start
2.middle
3. end
enter the place to delete the item: 2
enter the position to delete an item: 4
deleted item is: 60
enter your choice: 2
1. start
2. middle
3. end
enter the place to delete the item: 4
enter your choice: 3

start: 20
  30
  50
last: 70

enter your choice: 2
1. start
2. middle
3. end
derent the place to delete the item: 2
derent the position to delete an item: 3
deleted item is: 50
derent your choice: 2
1. start
2. middle
3. end
derent the place to delete the item: 2
derent the position to delete an item: 3
deleted item is: 50
derent your choice: 2
1. start
2. middle
3. end
derent the place to delete the item: 2
derent the position to delete an item: 1
deleted item is: 30
derent your choice: 2
1. start
2. middle
3. end
derent the place to delete the item: 1
deleted item is: 20
derent your choice: 3
last: 70
enter your choice: 2
1. start
2. middle
3. end
enter the place to delete the item: 1
deleted item is : 70
enter your choice: 3
*** LIST IS EMPTY ***
enter your choice: 2
*** LIST IS EMPTY ***
enter your choice: 8

consclusion: the program is error free

VIVA QUESTIONS:
1) List out the types of linked lists?
   Ans: i) circular linked lists  ii) doubly linked lists, iii) circular doubly linked list
2) What are the various operations performed on the linked lists?
   Ans: i) creating a list, ii) traversing the list iii) inserting an item etc.,
3) Another name for doubly linked list?
   Ans: two-way linked list.

Objective 31:
Program that implement stack and its operation by using the arrays

**Description:**

In this program we have to implement the stack operation by using the arrays. Here they stack operation are push and pop. Push operation is used to insert the elements into a stack and pop operation is used to remove the elements in to a stack.

**ALGORITHM FOR INSERTING AN ELEMENT IN A STACK:**

Function Push(s,top,x)

Step 1: [Check for stack overflow]
   If top>=n
      Then printf(“stack overflow”)
      Return

Step 2: [Increment Top]
   Top<-top+1

Step 3: [Insert element]
   S[top]<-x

Step 4: [finished]
   Return

**ALGORITHM FOR DELETING AN ELEMENT FROM A STACK:**

Function POP(s,top)

Step 1: [Check for stack underflow]
   If top=0
      Then printf(“stack underflow”)
      Exit

Step 2: [Decrement Top]
   Top<-top-1

Step 3: [Return former top element of stack]
   Return(S[top+1])

Step 4: [finished]
   Return

Flowchart:
Program:
# include <stdio.h>
# define size 4
int choice,top=0,a[size],item;
main()
{
 clrscr();
 while(1)
 {
   printf(" *** MENU ***
 1. PUSH
 2. POP
 3. TRAVERSE
 4. EXIT
");
 printf("enter your choice from menu:");
 scanf("%d",&choice);
 switch(choice)
 {
   case 1:push();
   break;
   case 2:pop();
   break;
   case 3:traverse();
   break;
   case 4:exit();
   break;
   default:printf("wrong choice
");
 }
 }
 getch();
}
push()
{
 if(size==top)
 printf("*** stack is full ***");
 else
 {
   printf("enter the item to be pushed into the stack:");
   scanf("%d",&item);
   top++;
   a[top]=item;
 }
}
pop()
{
 if(top==0)
 printf("*** stack is empty ***");
 else
 {
   }
item=a[top];
top--;
printf("the deleted item from stack is %d\n",item);
}
}
traverse()
{
    int i;
    if(top==0)
        printf("**** stack is empty ****");
    else
    {
        printf("**** stack display ***\n");
        for(i=1;i<=top;i++)
            if(i==top)
                printf("%d at %d ->top\n",a[i],i);
            else
                printf("%d at %d\n",a[i],i);
    }
}

Input/Output:

*** MENU ***
1. PUSH
2. POP
3. TRAVERSE
4. EXIT
enter your choice from menu:1
enter the item to be pushed into the stack:11
*** MENU ***
1. PUSH
2. POP
3. TRAVERSE
4. EXIT
enter your choice from menu:1
enter the item to be pushed into the stack:12
*** MENU ***
1. PUSH
2. POP
3. TRAVERSE
4. EXIT
enter your choice from menu:1
enter the item to be pushed into the stack:13
*** MENU ***
1. PUSH
2. POP 
3. TRAVERSE 
4. EXIT

enter your choice from menu: 1
enter the item to be pushed into the stack: 14

*** MENU ***
1. PUSH 
2. POP 
3. TRAVERSE 
4. EXIT

enter your choice from menu: 1
*** stack is full ***

*** MENU ***
1. PUSH 
2. POP 
3. TRAVERSE 
4. EXIT

enter your choice from menu: 3
*** stack display ***
11 at 1  
12 at 2  
13 at 3  
14 at 4 ->top 

*** MENU ***
1. PUSH 
2. POP 
3. TRAVERSE 
4. EXIT

enter your choice from menu: 2
the deleted item from stack is 14

*** MENU ***
1. PUSH 
2. POP 
3. TRAVERSE 
4. EXIT

enter your choice from menu: 2
the deleted item from stack is 13

*** MENU ***
1. PUSH 
2. POP 
3. TRAVERSE 
4. EXIT

enter your choice from menu: 2
the deleted item from stack is 12

*** MENU ***
1. PUSH
2. POP
3. TRAVERSE
4. EXIT

enter your choice from menu:2
the deleted item from stack is 11

*** MENU ***
1. PUSH
2. POP
3. TRAVERSE
4. EXIT

enter your choice from menu:2
*** stack is empty ***

*** MENU ***
1. PUSH
2. POP
3. TRAVERSE
4. EXIT

enter your choice from menu:3
**** stack is empty ****

*** MENU ***
1. PUSH
2. POP
3. TRAVERSE
4. EXIT

enter your choice from menu:4

**conclusion:** the program is error free

**VIVA QUESTIONS:**

1) Define Stack ?
   Ans: A stack is a linear data structure in which a data item is inserted and deleted at one end

2) Define data structure ?
   **Ans:** A data structure is a collection of organized data that are related to each other

3) What are the various operation performed on the stack ?
   Ans: push(), pop()

**Objective 32:**
Program that implement stack operations by using the pointers.

**Description:**
In this program we have to implement the stack operation by using the pointers. Here they stack operation are push and pop. Push operation is used to insert the elements into a stack and pop operation is used to remove the elements in to a stack.

**Algorithm:**
Step 1: Start
Step 2: Declare the structure for the stack pointers.
Step 3: Define the push function
Step 4: Define the pop function
Step 5: Define the display function
Step 6: Read the choice
Step 7: if choice = push
    Create a cell for the TOP cell in the stack.
    Place the date in the TOP cell
    Place the TOP pointer to the new cell
Step 8: if choice = pop
    Check if empty stack. If so, print stack is empty.
    Otherwise, remove the TOP cell.
Step 9: if choice = display
    Display all the elements in the Stack.
Step 10: Stop
Define structure for st_point pointer

Define push()

Define pop()

Define display()

Read choice, num1=0, num2=0

While(1)

Read choice

Switch choice

Push()

Pop()

Display()

Exit()

break

break

break

break

stop
PUSH(
)
Read
element, st_point
er
M=(struct st_point*)malloc(sizeof(struct st_point))
M -> elec=j
M -> l=t
T=m
Return to main program

Pop(
)
F
If
T
t=NULL
I = t->ele
T = t -> l
Return i
Print stack is empty
Return to main program
Program:

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

struct st_point
{
    int ele;
    struct st_point *l;
}

*t;

int i;
```

Display ()

Define st_point * pointer= NULL

Pointer = t

While(pointer! = NULL)

Print the element

Pointer=pointer->l

Return to main program
void push_ele(int j);
int pop_ele();
void display_ele();

void main()
{
    char choice,num1=0,num2=0;
    int i;
    while(1)
    {
        clrscr();
        printf("=================================================================================================");
        printf("\n\t MENU ");
        printf("=================================================================================================");
        printf("\n[1] Using Push Function ");
        printf("\n[2] Using Pop Function ");
        printf("\n[3] Elements present in Stack ");
        printf("\n[4] Exit\n");
        printf("\n\tEnter your choice: ");
        fflush(stdin);
        scanf("%c",&choice);
        switch(choice-'0')
        {
        case 1:
        {
            printf("\n\tElement to be pushed: ");
            scanf("%d",&num1);
            push_ele(num1);
            break;
        }
        case 2:
        {
            num2=pop_ele(1);
            printf("\n\tElement to be popped: %d\n\t",num2);
            getch();
            break;
        }
        case 3:
        {
            printf("\n\tElements present in the stack are:\n\t");
            display_ele();
            getch();
        }
break;
}

case 4:
    exit(1);
    break;

default:
    printf("Your choice is invalid.\n");
    break;
}
}

/*Inserting the elements using push function*/
void push_ele(int j)
{
    struct st_point *m;
    m=(struct st_point*)malloc(sizeof(struct st_point));
    m->ele=j;
    m->l=t;
    t=m;
    return;
}

/*Removing the elements using pop function*/
int pop_ele()
{
    if(t==NULL)
    {
        printf("STACK is Empty.\n");
        getch();
        exit(1);
    }
    else
    {
        int i=t->ele;
        t=t->l;
        return (i);
    }
    return 0;
}

/*Displaying the elements */
void display_ele()
{
    struct st_point *pointer=NULL;
pointer=t;
while(pointer!=NULL)
{
    printf("%d\t",pointer->ele);
    pointer=pointer->l;
}

Output:

================================================

MENU

[1] Using Push Function
[2] Using Pop Function
[3] Elements present in Stack
[4] Exit

Enter your choice: 1

Element to be pushed:23

================================================

MENU

[1] Using Push Function
[2] Using Pop Function
[3] Elements present in Stack
[4] Exit

Enter your choice: 3

Elements present in the stack are:
23

================================================

MENU

[1] Using Push Function
[2] Using Pop Function
[3] Elements present in Stack
[4] Exit

Enter your choice: 2

Element to be popped: 23
MENU

[1] Using Push Function
[2] Using Pop Function
[3] Elements present in Stack
[4] Exit

Enter your choice: 4
Exit the program

VIVA QUESTIONS:
1) Define Stack ?
Ans: A stack is a linear data structure in which a data item is inserted and deleted at one end

2) Define data structure ?
Ans: A data structure is a collection of organized data that are related to each other

3) What are the various operation performed on the stack ?
Ans: push(), pop()
Objective 33: 
Program that implement Queue operation by using the arrays

Description: In this program we have to implement the Queue operation by using the arrays. Here they Queue operation are push and pop. Push operation is used to insert the elements into a Queue and pop operation is used to remove the elements in to a Queue.

ALGORITHM FOR INSERTING AN ELEMENT IN TO A QUEUE:

Function QINSERT(Q,F,R,N,Y)
Step 1: [overflow]
   If R>=N
      Then printf(" overflow")
      Return
Step 2: [Increment rear pointer]
   R<-R+1
Step 3: [ Insert element]
   Q[R]<-y
Step 4: [Is front pointer properly set?]
   If F=0
      Then f<-1
      Return

ALGORITHM FOR DELETING AN ELEMENT FROM A STACK:

Function QDELETE(Q,F,R)
Step 1: [Underflow]
   If F=0
      Then printf("Queue underflow")
      Return(0)
Step 2: [Delete element]
   y<-q[f]
Step 3: [Is Queue Empty?]
   If F=R
      Then F=R=0
   Else
      F=F+1
Step 4:[Return element]
   Return(r)
Flowchart:

1. **Start**
   - Print the main menu
   - Read choice

2. **If choice==1**
   - Yes: Insert()
   - No: next choice

3. **If choice==2**
   - Yes: Delete()
   - No: next choice

4. **If choice==3**
   - Yes: View()
   - No: next choice

5. **If choice==4**
   - Yes: Exit()
   - No: Stop

---

**Insertion()**

1. **Start**
   - Insertion

2. **If front=max**
   - Yes: print Queue is full
   - No: print q[rear++]=a

3. **Stop**

---

**Deletion()**

1. **Start**
   - Deletion

2. **If front==rear**
   - Yes: print Queue is empty
   - No: print q[front++]

3. **Stop**

---

**Display()**

1. **Start**
   - View

2. **If front==rear**
   - Yes: print Queue is empty
   - No: print q[front++]

3. **Stop**
Program:

```c
#include <stdio.h>
define size 4
int front=0, rear=-1, item, choice, a[size];
main()
{
clrscr();
while(1)
{
printf("*** MENU ***
 1. INSERTION
 2. DELETION
 3. TRAVERSE
 4. EXIT
");
printf("enter your choice:");
scanf("%d", &choice);
switch(choice)
{
  case 1: insertion();
  break;
  case 2: deletion();
  break;
  case 3: traverse();
  break;
  case 4: exit();
  default: printf("*** wrong choice ***\n");
}
}
getch();
}
insertion()
{
  if (rear == size-1)
  printf("*** queue is full ***\n");
  else
  {
    printf("enter item into queue:");
    scanf("%d", &item);
    rear++;
    a[rear]=item;
  }
}
deletion()
{
  if (front == rear+1)
  printf("*** queue is empty ***\n");
  else
  {
```
item=a[front];
front++;
printf("the deleted item from queue is %d\n",item);
}
}
traverse()
{
    int i;
    if(front==rear+1)
        printf("*** queue is empty ***\n");
    else
    {
        for(i=front;i<=rear;i++)
            if(i==front && rear==i)
                printf("%d at %d ->front=rear\n",a[i],i);
            else
                if(i==rear)
                    printf("%d at %d ->rear\n",a[i],i);
                else
                    if(i==front)
                        printf("%d at %d ->front\n",a[i],i);
                    else
                        printf("%d at %d\n",a[i],i);
    }
}

Input/Output:

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT
enter your choice:1
enter item into queue:11
*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT
enter your choice:1
enter item into queue:12
*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT

enter your choice: 1
enter item into queue: 13

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT

enter your choice: 1
enter item into queue: 14

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT

enter your choice: 1

*** queue is full ***

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT

enter your choice: 3
11 at 0 ->front
12 at 1
13 at 2
14 at 3 ->rear

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT

enter your choice: 2
the deleted item from queue is 11

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT

enter your choice: 2
the deleted item from queue is 12

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE

4. EXIT
enter your choice:2
the deleted item from queue is 13

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT
enter your choice:2
the deleted item from queue is 14

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT
enter your choice:2
*** queue is empty ***

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT
enter your choice:3
*** queue is empty ***

*** MENU ***
1. INSERTION
2. DELETION
3. TRAVERSE
4. EXIT
enter your choice:4

**conclusion:** the program is error free

**VIVA QUESTIONS:**
1) Define queue ?
   **Ans:** A queue is a linear, sequential list of that are accessed in the order first in first out (FIFO).
2) Define circular queues ?
   **Ans:** A queue can also be circular in which case, it is called as a circular queue
3) What are the various stack oriented notations ?
   **Ans:** i) infix ii) prefix iii) postfix
Objective 34:
Program that implement Queue operation by using the pointers

Description:
In this program we have to implement the Queue operation by using the pointers. Here they Queue operation are push and pop. Push operation is used to insert the elements into a Queue and pop operation is used to remove the elements in to a Queue.

Algorithm:
Step 1: Start
Step 2: define structure for queue
Step 3: read choice
Step 4: if choice = insert
  i) read the element
  ii) create a data structure
  iii) if empty queue then front of queue pinter points to newly created data structure
  iv) otherwise end of the queue points to newly created data structure
Step 5: if choice = remove
  i) check if queue is empty . if so, print queue is empty
  ii) otherwise read the element pointed by front of the queue temp pointer points to front of queue
  iii) front of queue points to next element
  iv) free the element pointed by temp pointer
  v) return the element
  vi) print the element
Step 6: if choice = display
  i) check of empty queue if so, print queue empty
  ii) otherwise print the elements from front of the queue until the end of the queue
step 7: if choice = exits stop
Flowchart:

1. Define the structure for queue
2. Read the choice
3. If choice=insert
   - Read the element to be inserted
   - Queue->ele=ele
   - Queue->n=NULL
4. If !e_que()
   - Print queue is empty
5. If choice=remove
   - If !e_que()
     - Print queue is empty
   - If f_ptr=NULL
     - F+ptr=queue
   - If ptr->n!=NULL
     - Ptr=f_ptr
     - Ptr=ptr->n
   - Print queue is empty
6. If choice=display
   - If f_ptr=NULL
     - F+ptr=queue
   - If ptr->n!=NULL
     - Ptr=f_ptr
     - Ptr=ptr->n
   - Print invalid choice
7. If choice=exit
   - Print invalid choice
   - stop
8. Print j
program:
#define true 1
#define false 0

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

struct q_point
{
    int ele;
    struct q_point* n;
};

struct q_point *f_ptr = NULL;

int e_que(void);
void add_ele(int);
int rem_ele(void);
void show_ele();

/*main function*/
void main()
{

int ele, choice, j;
while(1)
{
    clrscr();
    printf("****IMPLEMENTATION OF QUEUE USING
POINTERS****");
    printf("="*36n") ;
    printf("\t\tMENU\n") ;
    printf("="*36n") ;
    printf("\n\t[1] To insert an element");
    printf("\n\t[2] To remove an element");
    printf("\n\t[3] To display all the elements");
    printf("\n\t[4] Exit");
    printf("\nEnter your choice:");
    scanf("%d", &choice);

    switch(choice)
    {
    case 1:
      {
       printf("\n\tElement to be inserted:");
       scanf("%d", &ele);
       add_ele(ele);
       getch();
       break;
      }
    case 2:
    {
     if(!e_que())
     {
      j=rem_ele();
      printf("\n%d is removed from the queue",j);
      getch();
     }
     else
     {
      printf("\n\tQueue is Empty.");
      getch();
     }
    break;
    }
    case 3: ;
show_ele();
getch();
break;

case 4:
    exit(1);
    break;

default:
    printf("\n\tInvalid choice.\n");
    getch();
    break;
}
}
}

/* Function to check if the queue is empty*/
int e_que(void)
{
    if(f_ptr==NULL)
        return true;
    return false;
}

/* Function to add an element to the queue*/
void add_ele(int ele)
{
    struct q_point *queue = (struct q_point*)malloc(sizeof(struct q_point));
    queue->ele = ele;
    queue->n = NULL;
    if(f_ptr==NULL)
        f_ptr = queue;
    else
    {
        struct q_point* ptr;
        ptr = f_ptr;
        for(ptr=f_ptr ;ptr->n!=NULL; ptr=ptr->n);
        ptr->n = queue;
    }
}

/* Function to remove an element from the queue*/
int rem_ele()
{
    struct q_point* queue=NULL;
if(e_que()==false)
{
    int j = f_ptr->ele;
    queue=f_ptr;
    f_ptr = f_ptr->n;
    free (queue);
    return j;
}
else
{
    printf(" Queue is empty.");
    return -9999;
}

/* Function to display the queue*/
void show_ele()
{
    struct q_point *ptr=NULL;
    ptr=f_ptr;
    if(e_que())
    {
        printf(" QUEUE is Empty.");
        return;
    }
    else
    {
        printf(" Elements present in Queue are:
        while(ptr!=NULL)
        {
            printf("%d\t",ptr->ele);
            ptr=ptr->n;
        }
    }
}

Output:

****IMPLEMENTATION OF QUEUE USING POINTERS****
=====================================================================

MENU
=====================================================================

[1] To insert an element
[2] To remove an element
[3] To display all the elements
Exit

Enter your choice: 1

Element to be inserted: 23

****IMPLEMENTATION OF QUEUE USING POINTERS****

==============================================

MENU

[1] To insert an element
[2] To remove an element
[3] To display all the elements
[4] Exit

Enter your choice:

3

Elements present in Queue are:

23

****IMPLEMENTATION OF QUEUE USING POINTERS****

==============================================

MENU

[1] To insert an element
[2] To remove an element
[3] To display all the elements
[4] Exit

Enter your choice: 2

23 is removed from the queue

****IMPLEMENTATION OF QUEUE USING POINTERS****

==============================================

MENU

[1] To insert an element
[2] To remove an element
[3] To display all the elements
[4] Exit

Enter your choice: 4

Exit

Conclusion: the program is error free
VIVA QUESATIONS:
1) Define queue?
Ans: A queue is a linear, sequential list of that are accessed in the order first in first out (FIFO).
2) Define circular queues?
Ans: A queue can also be circular in which case, it is called as a circular queue
3) What are the various stack oriented notations?
Ans: i) infix ii) prefix iii) postfix
Objective 35: Program that uses the stack operation to perform the following

i) converting infix expression into postfix expression
ii) evaluating the postfix expression

Description:
In this program we have to covert the given infix expression to postfix expression and the finally evaluating that postfix expression. Here we made use of stack operations. The property of stacks is last in first out. i.e., the item that is inserted last will be the first item remove.

Algorithm:

Step 1. start
Step 2. first initialize the stack to be empty
Step 3. for each character in the input string
   If input string is an operand, append to the output
   if the input string is a left paranthesis , push it onto the stack
   else
   if stack is empty or the operator has higher priority than the operator on the top of stack or the top of the stack is opening parenthesis
   then
   push the operator onto the stack
   else
   pop the operator from the stack and append to the output
Step 4. if the input string is a closing parenthesis , pop operators from the stack and append the operators to the output until an opening parenthesis is encountered. Pop the opening parenthesis from the stack and discard it.
Step 5. if the end of the input string is encountered , then iterate the loop until the stack is not empty. Pop the stack and append the remaining input string to the output.
Step 6. stop
\[ 2 + \frac{5-8}{3} = 258 - 3/+ \]

Input 2  

Input 5  

Input 8  

Input pop 8,5  

Push result  

Input 3  

Input 1  

Input +  

-1  

2  

1  

2
Program:

```c
#include<stdio.h>
#include<ctype.h>
#include<string.h>
static char str[20];
int top=-1;
main()
{
char in[20],post[20],ch;
int i,j,l;
clrscr();
printf("enter the string");
gets(in);
l=strlen(in);
for(i=0,j=0;i<l;i++)
if(isalpha(in[i]))
post[j++]=in[i];
else
{
if(in[i]==')')
push(in[i]);
else if(in[i]=='(')
while((ch=pop())!='(')
post[j++]=ch;
else
{
while(priority(in[i])<=priority(str[top]))
post[j++]=pop();
push(in[i]);
}
}
while(top!=-1)
post[j++]=pop();
post[j]="\0";
printf("\n equivalent infix to postfix is:%s",post);
getch();
}
priority (char c)
{
switch(c)
{
case'+':
case'-': return 1;
```
case '*':
case '/':
    return 2;
case '$':
    return 3;
}
return 0;
}
push(char c)
{
    str[++top]=c;
}
pop()
{
    return(str[top--]);
}

**Input/Output:**

enter the string(a+b)-(c-d)*e/f

equivalent infix to postfix is:ab+cd-e*f/-

enter the stringa+b/c*d

equivalent infix to postfix is:abc/d*+

**Conclusion:** the program is error free.

**ii) ALGORITHM:**

**Step 1: Start**

Step 2: Assign top=-1

Step 3: Read the input expression

Step 4: for i=0; s[i]!='\0' in steps of 1
Step 5: If isdigit(ch)
Step 5.1: Push(ch)
Step 6: otherwise
Step 6.1: op1 = pop()
Step 6.2: op2 = pop()
Step 7: c = op2 + op1
Step 8: Push(c)
Step 9: c = op2 - op1
Step 10: Push(c)
Step 11: c = pow(op2, op1)
Step 12: Push(c)
Step 13: c = op2 / op1
Step 14: Push(c)
Step 15: Print the result
Step 16: Push(int x)
Step 17: Increment top by 1
Step 18: s1.item(s1.top3) = x
Step 19: pop()
Step 20: Read x
Step 21: x1 = s1.item[s1.top]
Step 22: s1.top —
Step 23: return x
Step 24: Stop
Flowchart:

\[ a + b = ab^+ \]

Push a

Pop a

Push +

Push b

Pop ab+
Program:

```c
#include<stdio.h>
#include<ctype.h>
int stk[10],top=0,op1,op2,i;
main()
{
char postexp[10];
clrscr();
printf("enter the postfix expression:");
gets(postexp);
for(i=0;postexp[i]!="\0";i++)
{
if(isdigit(postexp[i]))
push(postexp[i]-48);
else
{
op1=pop();
op2=pop();
switch(postexp[i])
{
case `+':push(op1+op2);
    break;
case `-':push(op1-op2);
    break;
case `*':push(op1*op2);
    break;
case `/':push(op1/op2);
    break;
case `%':push(op1%op2);
    break;
case `.':exit();
}
}
printf("the result of postfix expression is: %d",pop());
getch();
}

pop()
{
return(stk[top--]);
}
```
push(int x)
{
  top++;
  stk[top]=x;
}

**Input/Output:**

- enter the postfix expression: 234+-
- the result of postfix expression is: 5

**Conclusion:** the program is error free.

**VIVA QUESATIONs:**

1) Define Stack ?
Ans: A stack is a linear data structure in which a data item is inserted and deleted at one end
2) Define data structure ?
**Ans:** A data structure is a collection of organized data that are related to each other
3) What are the various operation performed on the stack ?
Ans: push(), pop()
**Objective 36:** Program to

i) create a binary tree of integers

ii) Traversing the above binary tree in preorder, inorder and post order

**Description:** The data structure tree is of non linear type. Binary tree is bit special. The property of a binary tree is the value at the root must be greater than the left node and less than the right node. The binary tree consist of almost two childrens. Tree is a recursive data structure and recursive programming techniques are popularly used in trees. A tree can can be traversed in three major ways

i) Inorder traversal: here left child is visited first followed by root and finally by right child.

ii) Preorder traversal: Here root is visited first followed by left child and finally by right child.

iii) Postorder traversal: Here left child is visited first followed by right child finally by the root.

**ALGORITHM:**

Step 1: Start

Step 2: Define a structure btree

Step 3: Type define struct btree as node

Step 4: While(tree), begin

Step 5: Print MENU

Step 6: Write your choice

Step 7: If choice=1

Step 8: Enter your no of nodes

Step 9: Read nodes n

Step 10: for i=1 to n in steps of 1 do

Step 11: Print enter item

Step 12: Read item

Step 13: Root =call create (root, item).end for

Step 14: if choice=2
Step 15: Read element to be deleted

Step 16: Call delete(root, item) end for

Step 17: If choice=3

Step 18: Call preorder(root)

Step 19: Call postorder(root)

Step 20: Call inorder(root)

Step 21: Break, end of switch

Step 22: Stop

For insert function

Step 1: Start

Step 2: If t= null

Step 3: Allocate memory to temp

Step 4: Temp->data =item

Step 5: Temp-> lc=null

Step 6: Temp->rc=null

Step 7: return t to main and end

Step 8: If item< (l->data)

Step 9: T->lc=call insert(e->lc, t)

Step 10: T->rc=call insert(e->lc,t)

Step 11: Return t

Step 12: Stop

For **DELETION** function

Step 1: Start
Step 2: \( x = d \)

Step 3: while \( x \neq \text{null} \)

Step 4: If \( x \to \text{data} = t \)

Step 5: Break

Step 6: Parent = \( x \)

Step 7: if \( t < x \to \text{data} \)

Step 8: \( t = t \to \text{lc} \)

Step 9: \( t = t \to \text{rc} \)

Step 10: If \( x \to \text{lc} \neq \text{null} \) \&\& \( x \to \text{rc} \neq \text{null} \)

Step 11: parent = \( x \)

Step 12: If parent = \( \text{null} \)

Step 13: parent = \( \text{null} \)

Step 14: parent = \( \text{null} \)

Step 15: If \( p \to \text{lc} = x \to \text{rc} \)

Step 16: If \( p \to \text{rc} = x \to \text{rc} \)

Step 17: While \( \text{insert} \to \text{lc} = \text{null} \)

Step 18: \( \text{Insert} = \text{insert} \to \text{la} \)

Step 19: \( x \to \text{data} = \text{insert} \to \text{data} \)

Step 20: \( x = \text{insert} \)

Step 21: Return \( d \)

Step 22: Stop

For INORDER function

Step 1: Start
Step 2: If t!=null
Step 3: Call inorder(t->lc)
Step 4: Write t->data
Step 5: Call inorder(t->rc)
Step 6: Stop

For POSTORDER function
Step 1: Start
Step 2: If t!=null
Step 3: Call postorder(t->lc)
Step 4: Call postorder(t->rc)
Step 5: Write data
Step 6: Stop

For PREORDER function
Step 1: Start
Step 2: If t!=null
Step 3: Write data
Step 4: Call postorder(t->lc)
Step 5: Call postorder(t->rc)
Step 6: Stop
Flowchart:

Start

print main menu

Read choice

if choice==1

yes -> insert

no ->

if choice==2

yes -> delete

no ->

if choice==3

yes -> inorder

no ->

if choice==4

yes -> preorder

no ->

if choice==4

yes -> postorder

no ->

if choice==4

yes -> exit

no ->

Stop
Program:

**Insert**
- Start
- if t=null yes
- if item< t->data
- t->lc=insert()
t->rc=insert()
- return t
- stop

**Delete**
- Start
- r=d
- while x!=null no
- if x->data=t yes
- parent=x
- if t<x->data yes
- t=t->lc
t=t->rc
- if x->lc==null &
  x->rc==null yes
- parent=x

**Inorder**
- Start
- inorder
- if t!=null yes
- write t->data
- call preorder(t->lc)
call preorder(t->rc)
- stop

**Preorder**
- Start
- preorder
- if t!=null no
- write t->data
call preorder(t->lc)
call preorder(t->rc)
- stop

**Postorder**
- Start
- postorder
- if t!=null yes
- call postorder(t->lc)
call postorder(t->rc)
- stop

**Program:**

```
if t==null
    temp->data=item
    temp->rc=null
return t
```

```
r=d
while x!=null no
    if x->data=t yes
    parent=x
    if t<x->data yes
    t=t->lc
t=t->rc
    if x->lc==null &&
    x->rc==null yes
    parent=x
```
#include<stdio.h>
#include<alloc.h>

struct bstnode
{
    int data;
    struct bstnode *lc,*rc;
}*root,*a[20],*b[20];
int top=-1,top1=-1,n,i;

main()
{
    int ch,ele;
    struct bstnode *t,*insert(),*pop();
    clrscr();
    t=root=NULL;
    while(1)
    {
        printf("n **** M E N U **** n");
        printf("1.INSERT
2.RECURSSIVE TRAVERSE
3.NON-RECURSIVE TRAVERSE
4.EXIT
");
        printf("Enter your choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1: printf("Enter how many elements u want to insert:");
            scanf("%d",&n);
            printf("Enter tree elements:");
            for(i=1;i<=n;i++)
            {
                scanf("%d",&ele);
                t=insert(t,ele);
            }
            break;
        case 2: /* RECURSSIVE TRAVERSE */
            if(t==NULL)
                printf("**** TREE IS EMPTY ****");
            else
            {
                printf("INORDER : ");
                inorder(t);
                printf("\nPREORDER : ");
                preorder(t);
                printf("\nPOSTORDER : ");
                postorder(t);
            }
            break;
        case 3: /* NON-RECURSSIVE TRAVERSE */
            break;
        case 4: exit(0);
        }
case 3: /* NON-RECURSSIVE TRAVERSE */
    if(t==NULL)
        printf("TREE IS EMPTY");
    else
    {
        printf("INORDER :");
        nrinorder(t);
        printf("PREORDER :");
        nrpreorder(t);
        printf("POSTORDER :");
        nrpostorder(t);
    }
    break;

case 4:
    exit();
}
}
}
}

struct bstnode *insert(struct bstnode *x,int y)
{
    if(x==NULL)
    {
        x=malloc(sizeof(struct bstnode));
        x->data=y;
        x->lc=NULL;
        x->rc=NULL;
    }
    else
    {
        if(y<x->data)
            x->lc=insert(x->lc,y);
        else
            x->rc=insert(x->rc,y);
        return(x);
    }
}
inorder(struct bstnode *x)
{
    if(x!=NULL)
    {
        inorder(x->lc);
        printf("%3d",x->data);
        inorder(x->rc);
    }
}

preorder(struct bstnode *x)
{ if(x!=NULL) {
    printf("%3d",x->data);
    preorder(x->lc);
    preorder(x->rc);
} }

postorder(struct bstnode *x) {
    if(x!=NULL) {
        postorder(x->lc);
        postorder(x->rc);
        printf("%3d",x->data);
    }
}
nrinorder(struct bstnode *x) {
    struct bstnode *[l];
    l=x;
    do
    { while(l!=NULL) {
        push(l);
        l=l->lc;
    }
    while(top>-1) {
        l=pop();
        printf("%d",l->data);
        if(l->rc!=NULL) {
            l=l->rc;
            break;
        }
        else
            l=NULL;
    }
} while(l!=NULL);
}
nrpreorder(struct bstnode *x) {
    struct bstnode *[l];
    l=x;
do
{
    printf("%d",l->data);
    if(l->rc!=NULL)
push(l->rc);
    l=l->lc;
    if(l==NULL&&top>-1)
l=pop();
}while(l!=NULL);
}
nrpostorder(struct bstnode *x)
{
    struct bstnode *l;
l=x;
do
{
    while(l!=NULL)
    {
        push(l);
        if(l->rc!=NULL)
        {
            push(l->rc);
            b[++top1]=l->rc;
        }
l=l->lc;
    }
do
{
    l=pop();
    if(l!=b[top1])
        printf("%3d",l->data);
    else
    {
        top1--;l=
        break;
    }
}while(top>-1);
}while(l!=NULL&&top>-1);
}
push(struct bstnode *y)
{
top+=1;
a[top]=y;
}
struct bstnode *pop()
{
return a[top--];
}

**Input/Output:**

Enter your choice
1. Insert  2. Delete  3. Traversal
Enter the element 92
Enter your choice
1. Insert  2. Delete  3. Traversal
Enter the element 26
Enter your choice
1. Insert  2. Delete  3. Traversal
Enter the element 12
Enter your choice
1. Insert  2. Delete  3. Traversal
Enter the element 123
Enter your choice
1. Insert  2. Delete  3. Traversal
Enter the element 135
Enter your choice
1. Insert  2. Delete  3. Traversal
Enter the element 128
Enter your choice
1. Insert  2. Delete  3. Traversal
3
InorderSequence:  12 26 92 123 128 135
Preorder sequence: 92 26 12 123 135 128
Postorder sequence: 12 26 128 135 12 92
**Conclusion:** the program is error free.

**VIVA QUESATIONS:**

1) Define Binary tree ?
Ans: Binary tree is a bit special, because when they are in the sorted form, they facilitate quick search, insertion, and deletion.
2) How many ways a tree can be traversed ?
Ans: In three ways. They are i) In-order  ii) pre-order iii) post-order
3) define graph ?
Ans: A graph is a set of nodes(vertices) and a set of arcs(edges). A graph is connected if there is a path between any two nodes of the graph.

**Objective 37:**
To perform the linear search operation

**Description:** The linear search is most simple searching method. It does not expect the list to be sorted. The key which is to be searched is compared with each element of the list one by one. If a match exists, the search is terminated. If the end of list is reached it means that the search has failed and key has no matching in the list.

**ALGORITHM:**

**LINEAR SEARCH**

1. Start

2. Read the value of n

3. for i=1 to n increment in steps of 1
   
   Read the value of ith element into array

4. Read the element(x) to be searched

5. search <--linear(a,n,x)

6. if search equal to 0 goto step 7 otherwise goto step 8

7. print unsuccessful search

8. print successful search

9. stop

**LINEAR FUNCTION**

1. start

2. for i=1 to n increment in steps of 1

3. if m equal to k[i] goto step 4 otherwise goto step 2

4. return i

5. return 0

6. stop

**Flowchart:**
start

Read n

i=1
i<=n i++

Read a[i]

Read b

C=linear search(a,b)

If c! =0

Print c

T

F

Print element is not found

stop
\begin{verbatim}
for (i = 1; i < n; i++)
if (m == k[i])
  T
else
  F
end
\end{verbatim}
Program:

```c
#include<stdio.h>
main()
{
int i,j,n,a[10],key;
clrscr();
printf("enter range for array:");
scanf("%d",&n);
printf("enter elements into array:");
for(i=0;i<=n;i++)
  scanf("%d",&a[i]);
printf("enter the search element:");
scanf("%d",&key);
for(i=0;i<=n;i++)
  {
    if(key==a[i])
    {
      printf("element %d found at %d",key,i);
      break;
    }
    else
      if(i==n)
        printf("element %d not found in array",key);
  }
getch();
}
```

Input/Output:

enter range for array:4
enter elements into array:56
43
12
88
9
enter the search element:9
element 9 found at 4

enter range for array:5
enter elements into array:23
12
56
34
3
8
enter the search element:3
element 3 found at 4

**conclusion:** the program is error free
**VIVA QUESATIONS:**

1) Define linear search?
   Ans: The linear search is most simple serching method. It does not expect the list to be sorted. The key which is to be searched is compared with each element of the list one by one. If a match exists, the search is terminated. If the end of list is reached it means that the search has failed and key has no matching in the list.
Objective 38:

To perform the binary search operation

**Description:** Binary search is a vast improvement over the sequential search. For binary search to work, the item in the list must be in assorted order. The approach employed in the binary search is divide and conquer. If the list to be sorted for a specific item is not sorted, binary search fails.

**ALGORITHM:**

**BINARY SEARCH**

1. Start
2. Read the value of n
3. for i=1 to n increment in steps of 1
   Read the value of ith element into array
4. Read the element(x) to be searched
5. search<--binary(a,n,x)
6. if search equal to 0 goto step 7 otherwise goto step 8
7. print unsuccessful search
8. print successful search
9. stop

**BINARY SEARCH FUNCTION**

1. start
2. initialise low to 1 ,high to n, test to 0
3. if low<= high repeat through steps 4 to 9 otherwise goto step 10

4. assign (low+high)/2 to mid
5. if m<k[mid] goto step 6 otherwise goto step 7
6. assign mid-1 to high goto step 3
7. if m>k[mid] goto step 8 otherwise goto step 9
8. assign mid+1 to low
9. return mid
10. return 0
11.stop
Flowchart:

binarysearch

Start

Read value of n

for i=1;i<=n;i++

Read array element

search = binary (a, n, x)

if (Search==0)

print successful search

Stop

print unsuccessful search
binarysearch function

Start

initialise
low<-1, high<-1
test<-0

if(low<=high)

yes

mid<-(low+high)/2

if m<'(k+mid)

yes

high<-mid-1

no

no

if m>'(k+mid)

yes

low<-mid+1

no

Stop
**Program:**

```c
#include<stdio.h>
main()
{
int i,n,key,a[10],low,high,mid;
clrscr();
printf("enter range for array: ");
scanf("%d", &n);
printf("enter elements into array: ");
for(i=0;i<n;i++)
scanf("%d", &a[i]);
printf("the search element: ");
scanf("%d", &key);
low=0;
high=n-1;
for(i=0;i<n;i++)
{
mid=(low+high)/2;
if(a[mid]==key)
{
printf("element %d found at %d", key, mid);
because;
}
if(key<a[mid])
high=mid;
else
low=mid+1;
if(i==n-1)
printf("element %d not found in array", key);
}
getch();
}
```

**Input/Output:**

```
enter range for array: 4
enter elements into array: 12 23 34 45
the search element: 45
element 45 found at 3

enter range for array: 5
enter elements into array: 1 34 56 78 88
the search element: 45
element 45 not found in array
```

**conclusion:** the program is error free
1) Define Binary search?

**Ans:** Binary search is a vast improvement over the sequential search. For binary search to work, the item in the list must be in assorted order. The approach employed in the binary search is divide and conquer. If the list to be sorted for a specific item is not sorted, binary search fails.
Objective 39:
Program that implements the bubble sort method

Description: Bubble sort is the simplest and oldest sorting technique. This method takes two elements at a time. It compares these two elements. If the first element is less than the second one, they are left undisturbed. If the first element is greater than the second one, then they are swapped. The procedure continues with the next two elements until all the elements are sorted.

But bubble sort is an inefficient algorithm. The order of bubble sort algorithm is $O(n^2)$.

Algorithm:
i) Bubble Sort:
1. start
2. read the value of n
3. for i = 1 to n increment in steps of 1
   Read the value of ith element into array
4. call function to sort (bubble_sort(a, n))
5. for i = 1 to n increment in steps of 1
   print the value of ith element in the array
6. stop

Bubble Sort Function:
1. start
2. initialise last to n
3. for i = 1 to n increment in steps of 1
   begin
4. initialise ex to 0
5. for i = 1 to last-1 increment in steps of 1
   begin
6. if k[i] > k[i+1] goto step 7 otherwise goto step 5
   begin
7. assign k[i] to temp
   assign k[i+1] to k[i]
   assign temp to k[i+1]
   increment ex by 1
   end-if
   end inner for loop
11. if ex!=0
   assign last-1 to last
end for loop
12. stop
Flowchart:

**Start**

last = n

for (i = 1; i < n - 1; i++)

ex = 0

for (i = 1; i < last - 1; i++)

if *(i + 1) > *(i + i + 1)

    temp = *(i + 1)

    *(i + 1) = *(i + i + 1)

    *(i + i + 1) = temp

    ex++

    if ex != 0

        no

    yes

last = last - 1

stop

**Start**

Read n

for (i = 1; i < n; i++)

Read a(i)

call Bubble sort(a, n)

for (i = 1; i < n; i++)

print a(i)

stop
Program:

```c
#include<stdio.h>
main()
{
    int i,j,t,a[5],n;
    clrscr();
    printf("enter the range of array:");
    scanf("%d",&n);
    printf("enter elements into array:");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=0;i<n-1;i++)
        for(j=i+1;j<n;j++)
            if(a[i]>a[j])
            {
                t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
    printf("the sorted order is:");
    for(i=0;i<n;i++)
        printf("%d",a[i]);
    getch();
}
```

Input/Output:

enter the range of array:3
enter elements into array:3
2
1
the sorted order is: 1 2 3

enter the range of array:5
enter elements into array:56
23
34
12
8
the sorted order is: 8 12 23 34 56

conclusion: The program is error free

VIVA QUESATIONS
1) Define bubble sort?
Ans: Bubble sort is the simplest and oldest sorting technique. This method takes two elements at a time. It compares these two elements. If the first element is less than the second one, they are left undisturbed. If the first element is greater than the second one, then they are swapped. The procedure continues with the next two elements until all the elements are sorted.

2) Display the efficiency of bubble sort?
Ans: O(n²)
Program that implements the Quick sort method

**Description:** This method is invented by hoare, considered to be fast method to sort the elements. The method is also called partition exchange sorting. The method is based on divide and conquer technique. i.e., the entire list is divided into various partitions and sorting is applied again and again on the partition.

In this method the list is divided into two based on an element called pivot element. Usually the first element is considered to be the pivot element. Now move the pivot element to its correct position in the list. The elements to the left and pivot element are less that this while the elements to the right of pivot are greater than the pivot. The process is reapplied to each of these partitions till we got the sorted list of elements.

**Algorithm:**

**Quick Sort:**

1. start
2. if lowerbound < upperbound repeat through steps 3 to 13 otherwise goto step 14
   begin
3. assign lowerbound to i, upperbound to j, i to pivot
4. if i<j repeat through steps 5 to 10 otherwise goto step _
   Begin
5. if a[i]<=k[pivot] and i< upperbound repeat through step 6 otherwise goto step 7
   begin
6. assign i+1 to i
   end if
7. if k[j] > k[pivot] repeat through step 8 otherwise goto step 9
   begin
8. assign j-1 to j
   end if
9. if i< j goto step 10 other wise goto step 4
   Begin
10. call function to swap k[i] and k[j]
    end if
11. call function to swap k[pivot] and k[j]
12. call function qsort(x,lowerbound,j-1)
13. call function qsort(x,j+1,upperbound)
   end if
14. stop

**Flowchart:**
Program:
#include<stdio.h>
main()
{
    int x[10],i,n;
    clrscr();
    printf("enter no of elements:");
    scanf("%d",&n);
    printf("enter %d elements:",n);
    for(i=1;i<=n;i++)
        scanf("%d",&x[i]);
    quicksort(x,1,n);
    printf("sorted elements are:");
    for(i=1;i<=n;i++)
        printf("%3d",x[i]);
    getch();
}
quicksort(int x[10],int first,int last)
{
    int pivot,i,j,t;
    if(first<last)
    {
        pivot=first;
        i=first;
        j=last;
        while(i<j)
        {
            while(x[i]<=x[pivot] && i<last)
                i++;
            while(x[j]>x[pivot])
                j--;
            if(i<j)
            {
                t=x[i];
                x[i]=x[j];
                x[j]=t;
            }
        }
        t=x[pivot];
        x[pivot]=x[j];
        x[j]=t;
        quicksort(x,first,j-1);
        quicksort(x,j+1,last);
    }
}
**** OUTPUT ****

enter no of elements: 6
enter 6 elements: 23
12
45
34
21
87
sorted elements are: 12 21 23 34 45 87

conclusion: The program is error free

VIVA QUESTIONS

1) Define quick sort?
Ans:
This method is invented by hoare, considered to be fast method to sort the elements. The method is also called partition exchange sorting. The method is based on divide and conquer technique. i.e., the entire list is divided into various partitions and sorting is applied again and again on the partition.

In this method the list is divided into two based on an element called pivot element. Usually the first element is considered to be the pivot element. Now move the pivot element to its correct position in the list. The elements to the left and pivot element are less than this while the elements to the right of pivot are greater than the pivot. The process is reapplied to each of these partitions till we got the sorted list of elements.

2) Efficiency of quick sort?
Ans: O(n log n)

Objective 41: Program that implements the insertion sort method
**Description:** Insertion sort is similar to playing cards. To sort the cards in your hand you extract a card, shift the remaining cards, and then insert the extracted card in its correct place. The efficiency of insertion sort is $O(n^2)$.

**Algorithm:**

ii) Insertion Sort:
1. start
2. for $i = 1$ to $n$ increment in steps of 1 begin
   assign $k[i]$ to temp
3. for $j = i - 1$ down to $j > 0$ and $temp < k[j]$ begin
   assign $k[j]$ to $k[j+1]$ end inner for loop
4. assign temp to $k[j+1]$ end for loop
5. stop

**Flowchart:**
**INSERTION SORT**

Program:

```
class InsertionSort {
    public:
        void sort(int arr[], int n) {
            for(int i = 1; i < n; i++) {
                int temp = arr[i];
                int j = i - 1;
                while (j >= 0 && temp < arr[j]) {
                    arr[j + 1] = arr[j];
                    j--;
                }
                arr[j + 1] = temp;
            }
        }
};
```
```c
#include<stdio.h>
main()
{
    int i,j,t,a[10],n,p=0;
    clrscr();
    printf("enter the range of array:");
    scanf("%d",&n);
    printf("enter elements into array:");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=1;i<n;i++)
    {
        t=a[i];
        for(p=i;p>0 && a[p-1]>t;p--)
            a[p]=a[p-1];
        a[p]=t;
    }
    printf("the sorted order is:");
    for(i=0;i<n;i++)
        printf(" %d",a[i]);
    getch();
}

***** OUTPUT *****

enter the range of array:5
enter elements into array:5
4
3
2
1
the sorted order is:  1  2  3  4  5

enter the range of array:6
enter elements into array:23
12
89
45
67
34
the sorted order is:  12  23  34  45  67  89

**conclusion:** The program is error free

**VIVA QUESTIONS**
1) Define insertion sort?
**Ans:** Insertion sort is similar to playing cards. To sort the cards in your hand you extract a card, shift the remaining cards and then insert the extracted card in its correct place.

2) Efficiency of the insertion sort?
**Ans:** The efficiency of insertion sort is $O(n^2)$.
Program to implement the merge sort method

**Description:**

The merge sort splits the list to be sorted into two equal halves, and places them in separate arrays. Each array is recursively sorted, and then merged back together to form the final sorted list. Like most recursive sorts, the merge sort has an algorithmic complexity of \( O(n \log n) \).

**Algorithm:** main program

1. Start
2. declare the merge sort function
3. Declare the array and their size and initialize the j=0
4. read the array elements and then sort these elements.
5. read the array elements before the merge sort and then display the elements.
6. call the merge sort function
7. display the array elements after merge sort by using the following statement.
   ```
   for( j=0;j<Max_ary;j++)
   ```
8. Stop

**Subprogram**

1. initialize the array executing[MAX_ARY] and j=0,mid=0,mrg1=0,mrg2=0,size=start-end+1
2. check the condition if(end==start) then return
3. calculate the mid value
   ```
   Mid=(end+start)/2
   ```
4. call themerge_sort(x,end,mid)
5. merge_sort(x,mid+1,start)
6. performing the looping operation
   ```
   for(j=0;j<SIZE;j++) then its true
   ```
7. Executing[j]=x[end+1]
8. Mrg1=0;
9. calculate the mrg2=mid-end+1
10. performing the looping operation
    ```
    for(j=0;j<SIZE;j++) then its true then goto step9
    ```
11. check the condition
    1. if(mrg2<=start-end) is true goto ii). If not goto Step12.
    2. If(mrg1<=mid-end) is true goto iii). If not goto step11
    3. If(executing[mrg1]>executing[mrg2]) is true then follows.
       ```
       X[j+end]=executing[mrg2++]
       ```
    12. x[j+end]=executing[mrg1++]. If not goto Step11
13. x[j+end]=executing[mrg2++]
14. return to main program

**Flow chart:**
Start

Define merge_sort function

Declare the array and array size

Read the array elements

J=0
j++

j<MAX_ARY

Display the elements before merge sort

Mergesort(ary,0,MAX_ARY-1)

J=0
j++

j<MAX_ARY

Display the elements after merge sort

Stop
Merge_sort()
Size=start-end+1
Mid=0 Mrg1=0 Mrg2=0

If end==start
Mid=(end+start)/2
Call merge_sort(x,end,mid)
Call merge_sort(x,mid+1,start)

j=0
j++
j<size
Executing[j]=x[end+j]
Mrg1=0
True
j++
j<size
j=0
False
If mrg2<=start-end
False
X[j+end]=executing[mrg1++]
True
If mrg1<=mid-end
False
X[j+end]=executing[mrg2++]
False
If executing[mrg1]>executing[mrg2]
False
X[j+end]=executing[mrg1++]
True
Return to main program

X[j+end]=executing[mrg2++]
True
Program:
#include <stdio.h>
#include <stdlib.h>

#define MAX_ARY 10

void merge_sort(int x[], int end, int start);

int main(void) {
    int ary[MAX_ARY];
    int j = 0;

    printf("Enter the elements to be sorted: \n");
    for(j=0;j<MAX_ARY;j++)
        scanf("%d",&ary[j]);

    /* array before mergesort */
    printf("Before : ");
    for(j = 0; j < MAX_ARY; j++)
        printf(" %d", ary[j]);

    printf("\n");

    merge_sort(ary, 0, MAX_ARY - 1);

    /* array after mergesort */
    printf("After Merge Sort :");
    for(j = 0; j < MAX_ARY; j++)
        printf(" %d", ary[j]);

    printf("\n");
getch();
}

/* Method to implement Merge Sort*/
void merge_sort(int x[], int end, int start) {
    int j = 0;
    const int size = start - end + 1;
    int mid = 0;
    int mrg1 = 0;
    int mrg2 = 0;
    int executing[MAX_ARY];

    if(end == start)
        return;
mid = (end + start) / 2;
merge_sort(x, end, mid);
merge_sort(x, mid + 1, start);

for(j = 0; j < size; j++)
executing[j] = x[end + j];

mrg1 = 0;
mrg2 = mid - end + 1;

for(j = 0; j < size; j++) {
if(mrg2 <= start - end)
if(mrg1 <= mid - end)
if(executing[mrg1] > executing[mrg2])
x[j + end] = executing[mrg2++];
else
x[j + end] = executing[mrg1++];
else
x[j + end] = executing[mrg2++];
else
x[j + end] = executing[mrg1++];
}

Output:
Enter the elements to be sorted:
8 2 3 4 1 5 7 6 9 0
Before : 8 2 3 4 1 5 7 6 9 0
After Merge Sort : 0 1 2 3 4 5 6 7 8 9

Enter the elements to be sorted:
7 6 5 4 8 4 3 2 1 3
Before : 7 6 5 4 8 4 3 2 1 3
After Merge Sort : 1 2 3 3 4 4 5 6 7 8

Conclusion: the program is error free

VIVA QUESATIONS

1) Define merge sort?

Ans: The merge sort splits the list to be sorted into two equal halves, and places them in separate arrays. Each array is recursively sorted, and then merged back together to form the final sorted list.

2) Efficiency of merge sort?
Ans: $O(n \log n)$. 
Objective 43:
To implement the Lagrange interpolation and Newton Gregory forward interpolation

Lagrange Interpolation:

Algorithm:

Step 1. Read x, n

Step 2. for i = 1 to (n + 1) do Read xi, fi end for {the above statements reads x, s and the corresponding values of f is}

Step 3. Sum = 0

Step 4. for i = 1 to (n + 1) do

Step 5. Profvnc = 1

Step 6. for J = 1 to (n + 1) do

Step 7. If (j ≠ i) then profunc = profunc * (x - xj) / (xi - xj) end for

Step 8. Sum = Sum + fi * Profunc {sum is the value of f at x} end for

Step 9. Write x, sum

Step 10. STOP
Flow chart:

Start

Read n, y

i=0

is i<n

yes

Read x[i], f[i]

sum=0

i=0

is i<n

yes

pf=1

j=0

is j<n

yes

write

pf=pf*(y-x[i])/((x[j]-x[i]))

no

is j!=i

no

write

y, sum

stop
Program:

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

Main()
{
    Float y, x[20], f[20], sum, pf;
    Int I, j, n;
    Printf("enter the value of n");
    Scanf("%d", &n);
    Printf("enter the value to be found");
    Scanf("%f", &y);
    Printf("enter the values of xi’s & fi’s");
    For(i=0; i<n; i++)
    {
        Pf = 1;
        For(j=0; j<n; j++)
        {
            If(j!=i)
            {
                Pf *= (y-x[j])/(x[i] - x[j]);
            }
            Sum += f[i] * pf;
        }
    }
    Printf("x = %f ", y);
    Printf(" sum =%f ", sum);
}
```

Input/Output:

Enter the value of n 4
Enter the value to be found 2.5
Enter the values for xi’s & fi’s
1 1
2 8
3 27
4 64

X = 2.500000
Sum = 15.625000

Conclusion: The program is error free

VIVA QUESATIONS

1) Define storage class?
Ans: Storage class specifiers inform the compiler how to store the variable; the storage class specifiers in the C language are: auto, register, static, extern, typedef
Newton gregory forward interpolation.

Algorithm:

Step1: START

Step2: Read n

Step3: for i=0 to (n-1) do read xi,yi

Step4: read x

Step5: h←xi-x0

Step6: p←(x-x0)/n

Step7: for j=0 to n-2 do
   ∆1yj←yj+1-∆i-1

Step8: k←n-2

Step9: for i=2 to (n-1) do
   Step9.1: k←k-1
   Step9.2: for j=0 to k do
      ∆iyj←∆i-1 yj+1-∆i-1yj

Step10: Sumy←y0

Step11: Pvalue←1

Step12: Fact value←1

Step13: for l=1 to (n-1) do
   Step13.1: Pvalue←pvalue x (p-(l-1))
   Step13.2: factvalue←factvaluex1
   Step13.3: term←(pvalue x ∆ly) / factvalue
   Step13.4: Sumy←Sumy+term

Step14: Print x,SUMY

Step15: STOP

Flowchart:
Program:

Start

Read n

for i = 1 to n

Read x_i, y_i

Read x

h = x_i - x_0

p = (x - x_0)/n

for j = 1 to n-1

d1y_j = y_j + 1 - y_j

k = n - 2

for i = 2 to n-1

k = k - 1

d1y_j = d1y_j + d1y_{j-1}

sum_y = y_0

p_value = 1

fact_value = 1

for i = 1 to n-1

p_value = p_value * [p - 1]

fact_value = fact_value * 1

term = p_value * d1y_0 / fact_value

sum_y = sum_y + term

write x, sum_y

stop

C
#include<stdio.h>
#include<math.h>
Main()
{
    Int i, j, n, k, l;
    Float sumy, h, term, p, z, pvalue;
    Float x[25], y[25], d[25][25], factvalue;
    Printf("enter the value of n");
    Scanf("%d", &n);
    Printf("enter %d values for x, y\n", n);
    For(i=0; i<n; i++)
        Scanf("%f %f", &x[i], &y[i]);
    Printf("n enter z");
    Scanf("%f", &z);
    h = x[1] - x[0];
    p = (z - x[0]) / h;
    for(j=0; j<n-2; j++)
        d[i][j] = y[j+1] - y[j];
    k = n-2;
    for(i=2; i<n; i++)
    {
        k++;
        for(j=0; j<=k; j++)
            d[i][j] = d[i-1][j+1] - d[i-1][j];
    }
    For(l=1; l<n; l++)
    {
        pvalue *= (p - (l - 1));
        factvalue *= 1;
        Term = pvalue * d[l][0] / factvalue;
        Sumy += term;
    }
    Printf("n y value at z = %f is %f", z, sumy);
}
Enter n 7
Enter 7 data values for x, y

<table>
<thead>
<tr>
<th>Year</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1921</td>
<td>35</td>
</tr>
<tr>
<td>1931</td>
<td>42</td>
</tr>
<tr>
<td>1941</td>
<td>58</td>
</tr>
<tr>
<td>1951</td>
<td>84</td>
</tr>
<tr>
<td>1961</td>
<td>120</td>
</tr>
<tr>
<td>1971</td>
<td>165</td>
</tr>
<tr>
<td>1981</td>
<td>220</td>
</tr>
</tbody>
</table>

Enter z 1925
Y value at z = 1925.000000 is 36.756710
**Conclusion:** The program is error free

**VIVA QUESTIONS**

1) What is the use of goto statement?
**Ans:** The goto statement is used to alter the normal sequence of the program execution by unconditionally transferring control to some other part of the program.

2) What is the use of continue statement?
**Ans:** The continue statement is used to bypass the remainder of the current pass through a loop

**Objective 44:**
Implement in ‘C’ the linear regression and polynomial regression algorithms

**Linear regression**

**Algorithm:**

1. Read $n$
2. Sum$x$=0
3. Sum$x^2$=0
4. Sum$y$=0
5. Sum$x$$y$=0
6. for $i$=1 to $n$ do
   7. Read $x,y$
   8. Sum$x$=Sum$x$ + $x$
   9. Sum$x^2$=Sum$x^2$ + $x^2$
   10. Sum$y$=Sum$y$ + $y$
   11. Sum$x$$y$=Sum$x$$y$ + $x$$y$
    end for
8. $\text{denom}=n \times \text{Sum}x^2 - \text{Sum}x \times \text{Sum}x$
9. $a_0=(\text{Sum}y \times \text{Sum}x^2 - \text{Sum}x \times \text{Sum}x$$y$) / denom
10. $a_1=(n \times \text{Sum}x$$y$-Sum$x \times \text{Sum}y) / \text{denom}$
11. Write $a_1,a_0$
12. STOP
LINEAR REGRESSION

Start

Read n

sumx=0, sumy=0
sumxy=0, sumxsq=0

i=1

Read x, y

sumx=sumx+x
sumy=sumy+y
sumxy=sumxy+(x*y)
sumxsq=sumxsq+(x*x)

i=i+1

i > n

denom=(n*sumxsq-(sumx*sumx))

a0=(sumy*sumsq-sumx*sumxy)/denom
a1=(n*sumxy-sumx*sumy)/denom

write a0, a1

stop
Program:

```
#include<stdio.h>
#include<math.h>
Main()
{
    Int n, I;
    Float sumx, sumxsq, sumy, sumxy, x, y, a0, a1, denom;
    Printf("enter the n value");
    Scanf("%d", &n);
    Sumx = 0;
    Sumsq = 0;
    Sumy = 0;
    Sumxy = 0;
    For(i=0; i<n; i++)
    {
        Scanf("%f %f", &x, &y);
        Sumx += x;
        Sumsq += pow(x, 2);
        Sumy += y;
        Sumxy += x * y;
    }
    Denom = n * sumxsq - pow(sumx, 2);
    A0 = (sumy * sumxsq - sumx * sumxy) / denom;
    A1 = (n * sumxy - sumx * sumy) / denom;
    Printf("y = \%fx + \%fx", a1, a0);
}
```

Input/Output:

Enter the n value 7
1  2
2  5
4  7
5  10
6  12
8  15
9  19
Y = 1.980769x + 0.096154

Conclusion: The program is error free

VIVA QUESATIONS

1) What is the use of goto statement ?
   Ans: The goto statement is used to alter the normal sequence of the program execution by unconditionally transferring control to some other part of the program.

2) What is the use of continue statement ?
   Ans: The continue statement is used to bypass the remainder of the current pass through a loop
Polynomial regression

Algorithm:

Step 1: Start

Step 2: Read n

Step 3: Initialize sumx = 0, sumxsq = 0, sumy = 0, sumxy = 0, sumx3 = 0, sumx4 = 0, sumxsq = 0

Step 4: Initialize i=0

Step 5: Repeat steps 5 to 7 until i<n

Step 6: Read x,y

Step 7: Sumx = sumx + x
   Sumxsq = sumxsq + pow(x,2)
   Sumx3 = sumx3 + pow(x,3)
   Sumx4 = sumx4 + pow(x,4)
   Sumy = sumy + y
   Sumxy = sumxy + x*y
   Sumxsqy = sumxsqy + pow(x,2)*y

Step 8: Increment i by 1

Step 9: Assign
   a[0][0] = n
   a[0][1] = n
   a[0][2] = n
   a[0][3] = n
   a[1][0] = n
   a[1][1] = n
   a[1][2] = n
   a[1][3] = n
   a[2][0] = n
   a[2][1] = n
   a[2][2] = n
   a[2][3] = n

Step 10: Initialize i=0

Step 11: Repeat steps 11 to 15 until i<3
Step 12: Initialize j=0

Step 13: Repeat step 13 to 14 until j<=3

Step 14: Write a[i][j]

Step 15: Increment j by 1

Step 16: Increment I by 1

Step 17: Initialize k =0

Step 18: Repeat steps 18 to 27 until k<=2

Step 19: Initialize i=0

Step 20: Repeat step 20 to 26 until i<=2

Step 21: If I not equal to k

Step 22: Assign u=a[i][k]/a[k][k]

Step 23: Initialize j=k

Step 24: Repeat steps 24 and 25 until j<=3

Step 25: Assign a[i][j] = a[i][j] – u *a[k][j]

Step 26: Increment j by 1

Step 27: Increment i by 1

Step 28: Increment k by 1

Step 29: Initialize I =0

Step 30: Repeat steps 31 to 33 until i<3

Step 31: Assign b[i] = a[i][3]/a[i][i]

Step 32: Write I, b[i]

Step 33: Increment I by 1

Step 34: Write b[2],b[i],b[0]

Step 35: Stop
Program:

```c
#include<stdio.h>
#include<math.h>
main()
{
    Int n, i, j, k;
    Float sumx, sumxsq, sumy, sumxy, x, y;
    Float sumx3, sumx4, sumxsqy, a[20][20], u=0.0, b[20];
    Printf("Enter the n value");
    Scanf("%d", &n);
    Sumx = 0;
    Sumxsq = 0;
    Sumy = 0;
    Sumxy = 0;
    Sumx3 = 0;
    Sumx4 = 0;
    Sumxsqy = 0;
    For(i=0;  i<n; i++)
    {
        Scanf("%f %f", &x, &y);
        Sumx +=x;
        Sumxsq += pow(x,2);
        Sumx3 += pow(x,3);
        Sumx4 += pow(x,4);
        Sumy +=y;
        Sumxy += x * y;
        Sumxsqy += pow(x,2) *y;
    }
    A[0][0] = n;
    A[0][1] = sumx;
    A[0][2] = sumxsq;
    A[0][3] = sumy;
    A[1][0] = sumx;
    A[1][1] = sumxsq;
    A[1][2] = sumx3;
    A[1][3] = sumxy;
    A[2][0] = sumxsq;
    A[2][1] = sumx3;
    A[2][2] = sumx4;
    A[2][3] = sumxsqy;
    for(i=0;  i<3; i++)
    {
        for(j=0;  j<=3; j++)
            Printf("%0.2f",a[i][j]);
    }
    for(j=0;  i<3; j++)
        printf("%0.2f",a[i][j]);
```
Printf("n");
}
For(k=0; k<=2; k++)
{
    For(i=0;i<=2;i++)
    {
        If(i!=k)
            U=a[i][k]/a[k][k];
        For(j = k; j<=3; j++)
            A[i][j]=a[i][j] – u * a[k][j];
    }
}
For(i=0;i<3;i++)
{
    B[i] = a[i][3]/a[i][i];
    Printf("nx%d = %f", I, b[i]);
}
Printf("n");
Printf("y = %10.4fx +10.4 fx +%10.4f",b[2],b[1],b[0]);
}

Input/Output:

Enter the n value 10

<table>
<thead>
<tr>
<th></th>
<th>10.00</th>
<th>5.00</th>
<th>85.00</th>
<th>204.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5.00</td>
<td>85.00</td>
<td>125.00</td>
<td>513.00</td>
</tr>
<tr>
<td>3</td>
<td>85.00</td>
<td>125.00</td>
<td>1333.00</td>
<td>3193.00</td>
</tr>
</tbody>
</table>

X[0] = 2.030303
X[1] = 2.996970
X[2] = 1.984848
Y = 1.9848xsq + 2.9979x + 2.0303

Conclusion: The program is error free
VIVA QUESATIONS

1) Define insertion sort?
Ans: Insertion sort is similar to playing cards. To sort the cards in your hand you extract a card, shift the remaining cards, and then insert the extracted card in its correct place.

2) Efficiency of the insertion sort?
Ans: The efficiency of insertion sort is $O(n^2)$. 

**Objective 45:**
Program to Implement Trapezoidal and Simpson methods.

**Trapezoidal method:**

**Algorithm:**

Step 1. Read x1, x2, e \{ x1 and x2 are the two end points of the internal the allowed error in integral is e\}

Step 2. h = x2 - x1

Step 3. SI = (f(x1) + f(x2))/2;

Step 4. I = h - si

Step 5. i = 1 Repeat

Step 6. x = x1 + h/2

Step 7. for J = 1 to I do

Step 8. SI = SI + f(x)

Step 9. x = x + h

Endfor

Step 10. i = 2

Step 11. h = h/2 \{ Note that the internal has been halved above and the number of points where the function has to be computed is doubled\}

Step 12. i0 = i

Step 13. i1 = h * si

Step 14. until / I - i0 / <= e / i1 /

Step 15. Write I, h, i

Step 16. Stop
Flowchart:

**TRAPEZOIDAL RULE**

1. Start
2. Read a, b, n
3. \( h = \frac{(b-a)}{n} \)
4. \( k = a \)
5. \( \text{sum} = 0 \)
6. for \( i = 1 \) to \( n+1 \)
7. \( y[i] = (\exp(-\text{sqr} \ k)) \)
8. \( k = k + h \)
9. for \( i = 2 \) to \( n \)
10. \( \text{sum} = \text{sum} + y[i] \)
11. write ans
12. stop
Program:

```c
#include<stdio.h>
#include<math.h>
main()
{
    float h,a,b,n,x[20],y[20],sum=0,integral;
    int i;
    clrscr();
    printf("enter the value of a,b,n: ");
    scanf("%f %f %f",&a,&b,&n);
    printf("enter the values of x: ");
    for(i=0;i<=(n-1);i++)
    {
        scanf("%f",&x[i]);
    }
    printf("n enter the values of y: ");
    for(i=0;i<=(n-1);i++)
    {
        scanf("%f",&y[i]);
    }
    h=(b-a)/n;
    x[0]=a;
    for(i=1;i<=n-1;i++)
    {
        x[i]=x[i-1]+h;
        sum=sum+2*y[i];
    }
    sum=sum+y[b];
    integral=sum*(h/2);
    printf("approximate integral value is: %f",integral);
    getch();
}
```
**Input/Output:**

Enter the values of \(a, b, n\)

1  
2  
3  
Enter the values of \(x\):

1  
2  
3  
Enter the values of \(y\):

1  
2  
3  

Approximate integral value is 2.166667

**Conclusion:** The program is error free
Simpsons Method:

Algorithm:

Step 1. Read x1, x2, e

Step 2. h = (x2 - x1) / 2

Step 3. i = 2

Step 4. si = f(x1) + f(x2)

Step 5. s2 = 0

Step 6. s4 = f(x1 + h)

Step 7. I0 = 0

Step 8. In = (s + 4s4) * (h/3)

Repeat

Step 9. s2 = s2 + s4 \{s2 stores already computed functional value and s4 the value computed in the new nitration\}

Step 10. s4 = 0

Step 11. x = x1 + h/2

Step 12. for j = 1 to I do

Step 13. s4 = s4 + f(x)

Step 14. x = x + h

Step 15. h = h/2

Step 16. i = 2i

Step 17. Io = in

Step 18. in = (s1 + 2s2 + 4s4) * (h/3)

Step 19. until |In - Io| ≤ e

Step 20. Write In, h, i

Step 21. STOP
Flowchart:

**SIMPSON'S RULE**

```
Start

Read a, b, n

e = 0, f = 0

h = (b - a) / n

k = a

for i = 1 to n + 1

y[i] = (k * k * k) * exp(k - 1)

k = k + h

for i = 2 to n

is i mod 2 = 0

yes

f = f + y[i]

no

f = f + y[i]

e = e + y[i]

ans = h / 3 * (y[i] + y[n + 1] + 4 * e + 4 * f)

write ans

stop
```
Program:

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

main()
{
    float h,a,b,n,x[20],y[20],sum=0,itgl;
    int i;
    clrscr();
    printf("enter the values of a,b,n");
    scanf("%f%f%f",&a,&b,&n);
    printf("enter the values of x");
    for(i=0;i<=n;i++)
    {
        scanf("%f",&x[i]);
    }
    printf("n enter the values of y");
    for(i=0;i<=n;i++)
    {
        scanf("%f",&y[i]);
    }
    h=(b-a)/n;
    a=x[0];
    b=x[n];
    for(i=0;i<=(n-2);i++)
    {
        x[i]=x[i]+h;
        if(i%2==0)
        {
            sum=sum+4*y[i];
        }
        else
        {
            sum=sum+2*y[i];
        }
    }
    itgl=sum*(h/3);
    printf("integral value%f",itgl);
    getch();
}
Input/Output:

Enter the values of a, b, n
1
2
3
Enter the value of x
4
5
6
7
Enter the values of y
8
9
1
2
Integral value is 5.555556

Conclusion: The program is error free

VIVA QUESTIONS

1) Define Binary search?
Ans: Binary search is a vast improvement over the sequential search. For binary search to work, the item in the list must be in assorted order. The approach employed in the binary search is divide and conquer. If the list to be sorted for a specific item is not sorted, binary search fails.
**ADDITIONAL LAB EXPERIMENTS**

**Objective 46:** Write a C program for heap sort

**Description:** In this method, a tree structure named heap is used. A heap is a type of a binary tree. An ordered balanced binary tree is called a min-heap where the value at the root of any subtree is less than or equal to the value of either of its children. Heap sort is basically an improvement over the binary tree sort.

**Algorithm:**

**Heap sort**

SWAP FUNCTION

1. start

2. assign *a to temp

3. assign *b to *a

4. assign temp to *b

5. stop

HEAP SORT

1. start

2. assign n to i and a[n] to item

3. if i > 1 and a[i/2] < item repeat through step 4 other wise goto step 5
   begin

4. assign a[i/2] to a[i] and i/2 to i
   end if

5. assign item to a[i]

6. stop
Flowchart:

HEAP SORT

start

i=n
item=a[n]

if
"(k+j)>(k+pivot)"

no

a[i]=item

yes

a[i]=a[i/2]
i=i/2

stop
Program:
#include<stdio.h>
int a[20];
main()
{
int n,i;
clrscr();
printf("Enter number of elements: ");
scanf("%d",&n);
printf("Enter %d elements: ",n);
for(i=1;i<=n;i++)
    scanf("%d",&a[i]);
heapsort(n);
printf("Sorted elements are: 
");
for(i=1;i<=n;i++)
    printf("%3d",a[i]);
gech();
}
heapsort(int n)
{
int t;
while(n>1)
{
    maxheap(n);
    t=a[1];
a[1]=a[n];
a[n]=t;
n=n-1;
}
}
maxheap(int n)
{
int i,t,j;
for(i=2;i<=n;i++)
{
    t=a[i];
    j=i;
    while(a[j/2]<t&&j>1)
    {
        a[j]=a[j/2];
        j=j/2;
    }
    a[j]=t;
}
}
**Input/Output:**

Enter number of elements: 4
Enter 4 elements: 23
  4
  12
  8
Sorted elements are:
  4 8 12 23

Enter number of elements: 6
Enter 6 elements: 67
  23
  6
  45
  99
  78
Sorted elements are:
  6 23 45 67 78 99

**Conclusion:**

The program is error free

**VIVA QUESATIONS**

1) **Drawback of the binary tree?**
   Ans: Additional space is required for building the tree

2) The complexity of the heap sort algorithm?
   **Ans:** $O(n \log n)$
**Objective 47:**
Write a C program for selection sort

**Description:**
This is the simplest method of sorting. In this method, to sort the data in ascending order, the 0th element is compared with all other elements. If the 0th element is found to be greater than the compared element then they are interchanged.

**Algorithm:**

1. Start
2. Initialize the variables I, j, temp and arr[]
3. Read the loop and check the condition. If the condition is true, print the array elements and increment the I value. Else goto step 4
4. Read the loop and check the condition. If the condition true then goto next loop.
5. Read the loop and check the condition. If the condition true then goto if condition
6. If the condition if(arr[i] > arr[j]) is true then do the following steps
   i) temp = arr[i]
   ii) arr[i] = arr[j]
   iii) arr[j] = temp
7. Increment the j value
8. Perform the loop operation for the displaying the sorted elements.
9. Print the sorted elements
10. Stop
Initialize the $i, j, \text{temp}, \text{arr}[]$

$I = 0$

$I <= 4$

$I++$

Print the $\text{arr}[i]$ elements

$I = 0$

$I <= 3$

$I++$

$J = j + 1$

$J <= 4$

$J++$

If $\text{arr}[i] > \text{arr}[j]$

Temp = $\text{arr}[i]$

$\text{Arr}[i] = \text{arr}[j]$

$\text{Arr}[j] = \text{temp}$

Print $\text{arr}[i]$

Stop
Program:

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

Void main()
{

    Int i,j,temp;

    Clrscr();

    Printf("selection sort\n");

    Printf("array before sorting:\n");

    For(i=0;i<=3;i++)
    {
        Printf("%d	",arr[i]);
    }

    For(i=0;i<=3;i++)
    {
        For(j=j+1;j<=4;j++)
        {
            If(arr[i]>arr[j])
            {
                Temp=arr[i];
                Arr[i]=arr[j];
                Arr[j]=temp;
            }
        }
    }
```
Printf(“

 array after sorting:
”);

For(i=0;i<=4;i++)

Printf(“%d	”,arr[i]);

Getch();

Sample input & output:

1) Section sort
Array before sorting:
25   17  31        13     2
Array after sorting:
2    13    17    25    31
2) section sort
Array before sort
25   31  30 12   1
Array after sort
1   12     25  30   31

Conclusion: this program is error free

VIVA QUESTIONS

1) The complexity of the section sort algorithm?
   Ans: O(n^2)
2) 1) Drawback of the binary tree?
   Ans: Additional space is required for building the tree
3) The complexity of the heap sort algorithm?
   Ans: O(n log n)