List of Programs included

1. Insertion sort
2. Bubble sort
3. Selection sort
4. Circular Queue
5. Stack implementation using array
6. Linear search
7. Binary search (non recursive)
8. Fibonacci with recursion
9. GCD with recursion
10. Multiplication with recursion
11. Linear queue.
12. Binary search using recursion.
13. Stack implementation using linked list
14. Queue implementation using linked list
15. Linked list

****

1. **Insertion Sort :**

****

```
#include<iostream.h>
#include<conio.h>
class insertion
{
  private:
    int len;
    int *arr;
 public:
    insertion(int n)
    {
       len=n;
       arr=new int[n];
    }
    void enter()
    {
      cout<<"enter array"<<endl;
      for(int i=0;i<len;i++)
      {
     cout<<"enter a number at position"<<i<<"=";
     cin>>arr[i];
```

```
      }
    }
    void sort ()
    {
      for(int k=1;k<=len-1;k++)
      {
     int temp,j;
     temp=arr[k];
     j=k-1;
     while(temp<arr[j]&&j>=0)
     {
       arr[j+1]=arr[j];
       j=j-1;
     }
     arr[j+1]=temp;
      }
     }
     void display()
     {
       for(int i=0;i<len;i++)
       cout<<arr[i]<<" ";
     }
 };
 void main()
 {
    clrscr();
    int l;
    cout<<"enter the length of array = ";
    cin>>l;
    insertion obj(l);
    obj.enter();
    cout<<"array entered"<<endl;
    obj.display();
    obj.sort();
    cout<<"\n array sorted"<<endl;
    obj.display();
    getch();
}
```

****
2. **Bubble sort**
****

```
#include<iostream.h>
#include<conio.h>
class bubble
 {
 public:
 int n,i,a[20],ptr;
  void getvalue();
  void showvalue();
  void compare();
 };
     void bubble::getvalue()
     {
      cout<<"\nenter the no. of elements you want to enter(MAX
20)  ";
      cin>>n;
      cout<<"\n\nenter the elements in array";
       for(i=0;i<n;i++)
           {
       cout<<"\n\nenter the element "<<i+1<<" : ";
       cin>>a[i];
           }
     }

       void bubble::showvalue()
       {
      cout<<"\n\nelements in array are ";
      for(i=0;i<n;i++)
         {
       cout<<"\t"<<a[i];
         }
       }

      void bubble::compare()
      {
```

```
        for(i=0;i<n;i++)
      {   ptr=1;
         while(ptr <= n-i-1)
          {
          if( a[ptr]>a[ptr+1] )
              {
            int temp;
            temp= a[ptr];
            a[ptr]=a[ptr+1];
            a[ptr+1]=temp;

          }
          ptr=ptr+1;
        }
       }

       }



void main()
{
 clrscr();
 bubble  bb;
 bb.getvalue();
 bb.showvalue();
 bb.compare();
 cout<<"\n\nsorted elements"<<endl;
 bb.showvalue();


getch();
}
```

****

3.**Selection Sort**

****

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

```
#include <conio.h>
#define MAX 10

class selsort{
    int arr[MAX],n;
    public:
    void getdata();
    void showdata();
    void sortLogic();
};

void selsort :: getdata(){
    cout<<"How many elements you require : ";
    cin>>n;
    for(int i=0;i<n;i++)
        cin>>arr[i];
}

void selsort :: showdata(){
    cout<<"\n--Display--\n";
    for(int i=0;i<n;i++)
        cout<<arr[i]<<"   ";
}

void selsort :: sortLogic(){
    int temp,min;
    for(int i=0;i<n;i++){
        min=i;
        for(int j=i+1;j<n;j++){
            if(arr[min] > arr[j]){
                min=j;
            }
        }
        temp = arr[min];
        arr[min] = arr[i];
        arr[i] = temp;
        cout<<"\n arr[min] = "<<arr[min]<<"  arr[i] = "<<arr[i];
    }
}
```

```
void main(){
    clrscr();
    cout<<"\n*****Selection Sort*****\n";
    selsort obj;
    obj.getdata();
    obj.sortLogic();
    obj.showdata();
    getch();
}
```

****

4.**Circular queue**

****

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

  class circularqueue
    {
     private:
     int front,rear,size;
     int *elements;
     public:
     circularqueue()
     {
      elements=new int[10];
        size=10;
        front=-1;
        rear=-1;
      }
       circularqueue(int n)
       {
        elements=new int[n];
          size=n;
          front=-1;
```

```
  rear=-1;
}
  ~circularqueue()
  {
   delete elements;
  front=-1;
  rear=-1;
  size=0;
   }
  int isempty()
  {
  if(front==-1)
     return 1;
  else
     return 0;
   }
  int isfull()
  {
   if(((rear==size-1) && (front==0)) || (front==rear+1))
        return 1;
   else
        return 0;
   }
     void enqueue(int value)
      {
       int t;
       t=isempty();
       if(t==1)
         {
       rear=0;
       front=0;
         }
       else if(rear==size-1)
          {
            rear=0;
          }
       else
        rear++;
         elements[rear]=value;
```

```
    }
       int dequeue()
      {
        int temp;
        temp=elements[front];
          if(rear==front)
          {
           rear=-1;
           front=-1;
           }
          else if(front==size-1)
               front=0;
          else
             front++;
          return temp;
         }
       int peek()
       {
      return elements[front];
       }
  void display()
  {
    if(front<=rear)
    {
     for(int i=front;i<=rear;i++)
       cout<<elements[i]<<" ";
    }
      else
     {
      for(int j=front;j<=size-1;j++)
         cout<<elements[j]<<" ";
      for(int k=0;k<=rear;k++)
           cout<<"*"<<elements[k]<<" ";
      }
  }
 };

void main()
{
```

```
  clrscr();
 int s,ch,p,num;
   cout<<"Enter size of queue =";
      cin>>s;
  circularqueue obj(s);
 do
 {
  cout<<"\n.....MAIN MENU....."<<endl;
  cout<<"Press 1 to insert"<<endl;
  cout<<"Press 2 to delete"<<endl;
  cout<<"Press 3 to peek"<<endl;
  cout<<"Press 4 to display"<<endl;
  cout<<"Press 5 to exit"<<endl;
  cout<<"Enter your choice= ";
 cin>>ch;

switch(ch)
  {
 case 1:
      p=obj.isfull();
      if(p==1)
          cout<<"Queue is full";
      else
       {
        cout<<"Enter a number =";
           cin>>num;
        obj.enqueue(num);
       }
       break;
  case 2:
       p=obj.isempty();
         if(p==1)
           cout<<"Queue is empty";
         else
            cout<<"number deleted is = "<<
obj.dequeue();
       break;
  case 3:
       p=obj.isempty();
```

```
            if(p==1)
              cout<<"Queue is empty";
            else
              cout<<"number at front is = "<<obj.peek();
          break;
       case 4:
          p=obj.isempty();
             if(p==1)
                cout<<"Queue is empty";
             else
                obj.display();
       case 5:
          exit;
          break;

       default: cout<<"Entered wrong choice";

       }
       cout<<"\nPress any key to continue"<<endl;
      getch();
      clrscr();
      }while(ch!=5);
        getch();
        }
```

****

5.**Stack implementation**

****

```
#include<iostream.h>
#include<conio.h>
class stack
{
private:
struct node
{
int data;
node *link;
```

```
}*top;
public:
stack();
void push(int item);
int pop();

~stack();
};
stack::stack()
{
top=NULL;
}
void stack :: push(int item)
{
node *temp;
temp=new node;

if(temp==NULL)
cout<<endl<<"stack is full";

temp->data=item;
temp->link=top;
top=temp;
}

int stack::pop()
{
if(top==NULL)
{
cout<<endl<<"stack is empty";
return NULL;
}
node *temp;
int item;
temp=top;
item=temp->data;
top=top->link;
delete temp;
return item;
```

```
}

stack:: ~stack()
{
if(top==NULL)
return;
node *temp;
while(top!=NULL)
{
temp=top;
top=top->link;
delete temp;
}
}

void main()
{
stack s;
s.push(14);
s.push(-3);
s.push(18);
s.push(29);
s.push(31);
s.push(16);
int i=s.pop();
cout<<"item popped :"<<i;
i=s.pop();
cout<<endl<<"item popped:"<<i;

i=s.pop();
cout<<endl<<"item popped:"<<i;
getch();
}
```

****
6.Linear search
****

Linear Search

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

void main()
{
clrscr();

int a[100],i,n,item,s=0;

cout<<"\n------------ LINEAR SEARCH ------------ \n\n";
cout<<"Enter No. of Elements=";
cin>>n;

cout<<"\nEnter Elements=\n";
for(i=1;i<=n;i++)
{
cin>>a[i];
}

cout<<"\nEnter Element you want to Search=";
cin>>item;

for(i=1;i<=n;i++)                          //Array Elements
Comparsion with Item
{
if(a[i]==item)
{
cout<<"\nData is Found at Location : "<<i;
s=1;
break;
}
}

if(s==0)
{
cout<<"Data is Not Found";
}
```

```
getch();
}

**********************************************************************
Binary search
**********************************************************************

#include<iostream.h>
 #include<conio.h>

 class searching
 {
 private:
    int len;
    int *arr;
 public:
    searching(int m)
    {
      len=m;
      arr=new int[m];
     }
    void enter()
    {
      cout<<".....enter an array..."<<endl;
     for(int i=0;i<len;i++)
     {
     cout<<"enter number at position "<<(i)<<" = ";
        cin>>arr[i];
     }
     }

       void sea(int search)
       {

      int b,e,m;
       b=0;
       e=len-1;
```

```
        while(b<=e)
        {
           m=(b+e)/2;
             if(search==arr[m])
            {
            cout<<"found at position "<<m;
            break;
            }
             else if(search>arr[m])
             b=m+1;
             else
            e=m-1;
          };
           if(b>e)
           cout<<"not found";
           }
           };
      void main()
      {
        clrscr();
        int l,s;
        cout<<"enter length of array= ";
          cin>>l;
        searching obj(l);
          obj.enter();
         cout<<"enter number to be searched= ";
           cin>>s;
            obj.sea(s);
          getch();
        }
```

****

8.Fibonacci with recursion

****

```
#include<iostream.h>
#include<conio.h>
 class ayush
```

```
{
 public:
   int fib(int n)
    {
   if(n<=1)
     return n;
  else
     return (fib(n-1)+fib(n-2));
  }
 };
   void main()
   {
    clrscr();
   int a;
    ayush obj;
      cout<<"enter a number =";
         cin>>a;

       for(int i=0;i<a;i++)
       {
         cout<<obj.fib(i)<<" ";
       }
         getch();
  }
```

****

9.**GCD with recursion**

****

```
#include<iostream.h>
#include<conio.h>
class gcdimpl
{
 public:
   int gcd(int p,int q)
     {
      if(q==0)
      return p;
      else
```

```
      return gcd(q,p%q);
     }
 };
   void main()
     {
      clrscr();
     int a,b,result;
     cout<<"enter first number =";
       cin>>a;
     cout<<"enter second number =";
       cin>>b;
     if(a<b)
       {
        int t=a;
          a=b;
          b=a;
        }
       gcdimpl obj;
     result=obj.gcd(a,b);
       cout<<"The GCD of two numbers is "<<result;
      getch();
     }
```

****

10.**Multiplication with recursion**

****

```
#include<iostream.h>
#include<conio.h>
 class impl
 {
   public:
      int mul(int a,int b)
     {
      if(b==1)
         return a;
      else
         return (a+mul(a,b-1));
```

```
    }
};
  void main()
  {
   clrscr();
     int a,b,result;
     impl obj;
    cout<<"Enter first number =";
      cin>>a;
    cout<<"Enter second number =";
      cin>>b;
     result=obj.mul(a,b);
    cout<<"Multiplication of two numberis "<<result;
     getch();

  }
```

****

11.**Linear queue**

****

```
#include<iostream.h>
class queue
 {
  private:
    struct node
    {
     int data;
     node *link;
     }*front, *rear;

  public:
   queue();
   void addq(int item);
   int delq();
   ~queue();
 };
```

```
//initialise data member
queue::queue()
{
 front=rear=NULL;
}
//adds an element to the queue
void queue::addq(int item)
{
 node *temp;
 temp=new node;
 if (temp==NULL)
   cout<<"\nQueue is full";
   temp->data=item;
   temp->link=NULL;
   if (front==NULL)
     {
       rear=front=temp;
       return;
     }
     rear->link=temp;
     rear=rear->link;
 }

 //removes an element from the queue
 int queue::delq()
 {
   if (front==NULL)
      {
     cout<<"\n Qeue is empty";
     return NULL;
      }
   node *temp;
   int item;
   item=front->data;
   temp=front;
   front=front->link;
   delete temp;
   return item;
 }
```

```
//deallocates memory
queue::~queue()
{
 if (front==NULL)
   return;
   node *temp;
   while (front!=NULL)
   {
     temp=front;
     front=front->link;
     delete temp;
   }
}

void main()
{
   queue a;
   a.addq(11);
   a.addq(-8);
   a.addq(23);
   a.addq(19);
   a.addq(15);
   a.addq(16);
   a.addq(28);
   int i=a.delq();
   cout<<"\n item extracted:"<<i;
   i=a.delq();
   cout<<"\n item extracted: "<< i;
   i= a.delq();
   cout<<"\n item extracted:"<< i;
 }
```

****

12.**Binary search using recursion**

****

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

```
class recsea
{
 private:
   int len;
   int*arr;
 public:
   recsea(int n)
   {
    len=n;
    arr=new int[n];
    }
    void enter()
    {
     cout<<"enter array"<<endl;
    for(int i=0;i<len;i++)
    {
     cout<<"enter a number at position "<<i<<" =";
       cin>>arr[i];
    }
    }
     int search(int b,int e,int s)
     {
   int m;
    if(b<=e)
    {
     m=(b+e)/2;
       if(arr[m]==s)
          return m;
       else if(s>arr[m])
        return search(m+1,e,s);
       else if(s<arr[m])
        return search(b,m-1,s);
     }
     else if(b>e)
        return(-1);
      }
     };
     void main()
   {
```

```
      clrscr();
       int le,hi,lo,t,se;
          cout<<"enter size of array =";
            cin>>le;
         recsea obj(le);
          obj.enter();
            lo=0;
            hi=le-1;
           cout<<"enter number to be searched =";
            cin>>se;
           t=obj.search(lo,hi,se);
          if(t==(-1))
            cout<<"not found";
              else
              cout<<"found at position "<<t;
              getch();
          }
```

****

13. Stack implementation using linked list

****

```
#include<iostream.h>
#include<conio.h>
class stack
{
private:
struct node
{
int data;
node *link;
}*top;
public:
stack();
void push(int item);
int pop();

~stack();
};
stack::stack()
{
```

```
top=NULL;
}
void stack :: push(int item)
{
node *temp;
temp=new node;

if(temp==NULL)
cout<<endl<<"stack is full";

temp->data=item;
temp->link=top;
top=temp;
}

int stack::pop()
{
if(top==NULL)
{
cout<<endl<<"stack is empty";
return NULL;
}
node *temp;
int item;
temp=top;
item=temp->data;
top=top->link;
delete temp;
return item;
}

stack:: ~stack()
{
if(top==NULL)
return;
node *temp;
while(top!=NULL)
{
temp=top;
```

```
top=top->link;
delete temp;
}
}

void main()
{
stack s;
s.push(14);
s.push(-3);
s.push(18);
s.push(29);
s.push(31);
s.push(16);
int i=s.pop();
cout<<"item popped :"<<i;
i=s.pop();
cout<<endl<<"item popped:"<<i;

i=s.pop();
cout<<endl<<"item popped:"<<i;
getch();
}
```

****

14 Queue implementation using linked list

****

```
#include<iostream.h>
class queue
 {
  private:
    struct node
    {
     int data;
     node *link;
     }*front, *rear;

  public:
   queue();
   void addq(int item);
```

```
   int delq();
   ~queue();
 };
//initialise data member
queue::queue()
{
 front=rear=NULL;
}
//adds an element to the queue
void queue::addq(int item)
{
 node *temp;
 temp=new node;
 if (temp==NULL)
   cout<<"\nQueue is full";
   temp->data=item;
   temp->link=NULL;
   if (front==NULL)
     {
       rear=front=temp;
       return;
     }
     rear->link=temp;
     rear=rear->link;
 }

 //removes an element from the queue
 int queue::delq()
 {
   if (front==NULL)
      {
     cout<<"\n Qeue is empty";
     return NULL;
      }
   node *temp;
   int item;
   item=front->data;
   temp=front;
   front=front->link;
```

```
  delete temp;
  return item;
}

//deallocates memory
queue::~queue()
{
 if (front==NULL)
   return;
   node *temp;
   while (front!=NULL)
   {
     temp=front;
     front=front->link;
     delete temp;
   }
}

void main()
{
   queue a;
   a.addq(11);
   a.addq(-8);
   a.addq(23);
   a.addq(19);
   a.addq(15);
   a.addq(16);
   a.addq(28);
   int i=a.delq();
   cout<<"\n item extracted:"<<i;
   i=a.delq();
   cout<<"\n item extracted: "<< i;
   i= a.delq();
   cout<<"\n item extracted:"<< i;
 }
```