1. Write a program to implement Stack ADT using an Array.

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;
class Stack
{
private:
int stack[10];
int MaxCapacity;
int top;
public:
Stack()
{
MaxCapacity = 10;
top = -1;
}
void push(int Element);
int pop();
void display();
};
void Stack :: push(int Element)
{
if(top==MaxCapacity-1)
cout<<"Stack overflow";
else
stack[++top] = Element;
}
int Stack :: pop()
{
if(top==-1)
cout<<"Stack Underflow";
else
{
cout<<stack[top]<<"is popped out";
return(stack[top--]);
}
}
void Stack :: display()
{
cout<<"*****************"<<endl;
if(top==-1)
cout<<"Stack Underflow";
else
for(int i=top;i>=0;i--)
cout<<stack[i]<<endl;
cout<<"*****************"<<endl;
}
int main()
{
int ch,val;
Stack S;
while(1)
{
cout<<"\n1.push\n2.pop\n3.display\n4.exit\n";
cin>>ch;
```

```
switch(ch)
{
case 1: cout<<"Enter the element to push: ";
cin>>val;
S.push(val);
cout<<val<<"is added to stack"<<endl;
break;
case 2: S.pop();
break;
case 3: S.display();
break;
case 4: exit(0);
}
}
}
```

2. Write a program to implement Queue ADT using an Array.

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;
class Queue
{
private:
public:
int queue[10];
int rear,front;
int maxcapacity;
Queue()
{
rear=-1;
front=-1;
maxcapacity=10;
}
void insert(int Element);
void delet();
void display();
};
void Queue::insert(int Element)
{
if(rear>maxcapacity)
cout<<"Queue is Full";
else
{
queue[++rear]=Element;
front=0;
}
}
void Queue:: delet()
{
if(front==-1 && rear==-1)
cout <<"Queue is Empty";
else if(front==rear)
{
front=-1;
```

```
rear=-1;
}
else
front++;
}
void Queue::display()
{
cout<<"*******"<<endl;
if(front==-1 && rear==-1)
cout <<"Queue is Empty";
else
for(int i=front;i<=rear;i++)
{
if(front!=-1 && rear!=-1)
cout<<queue[i]<<endl;
}
}
int main()
{
int ch,val;
Queue q;
while(1)
{
cout<<"\n1.Insert\n2.Remove\n3.Display\n4.exit\n";
cin>>ch;
switch(ch)
{
case 1: cout<<"Enter the element to Add: ";
cin>>val;
q.insert(val);
cout<<val<<"is added to Queue"<<endl;
break;
case 2: q.delet();
break;
case 3: q.display();
break;
case 4: exit(0);
}
}
}
```

3. Write a program to create a Single Linked List.

```
#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class node

{

public:

int data;

node *next;
```

```
node(int d)
{
data=d;
}
};
class single
{
node *nn,*tn,*rn,*root,*tail,*t1,*t2;
public:
int nc;
single()
{
root=NULL;
nc=0;
}
void insertpos(int,int);
void deletepos(int);
void create(int);
void display();
};
void single::create(int no)
{
nn=new node(no);
nn->next=NULL;
nc++;
if(root==NULL)
{
root=tail=nn;
}
else
{
tail->next=nn;
tail=nn;
}
}
void single::display()
```

```
{
cout<<"\n number of nodes are: "<<nc;
cout<<endl<<"single linked list is: \n";
for(tn=root;tn!=NULL;tn=tn->next)
{
cout<<tn->data<<"->";
//cout<<"NULL";
}
cout<<"NULL";
}
void single::insertpos(int p,int no)
{
int i;
nn=new node(no);
nn->next=NULL;
if(p==1)
{
nn->next=root;
root=nn;
}
else if(p>nc)
{
tail->next=nn;
tail=nn;
}
else if(p>1 &&p<=nc)
{
i=2;
tn=root;
while(tn->next!=NULL && i<p)
{
tn=tn->next;
i++;
}
nn->next=tn->next;
tn->next=nn;
```

```
}
nc++;
}
void single::deletepos(int p)
{
int i;
if(p==1)
{
rn=root;
root=root->next;
}
else if(p==nc)
{
rn=tail;
tn=root;
while(tn->next!=tail)
{
tn=tn->next;
}
tn->next=NULL;
tail=tn;
}
else if(p>1 && p<nc)
{
tn=root;
i=2;
while(tn->next!=NULL && i<p)
{
tn=tn->next;
i++;
}
rn=tn->next;
tn->next=rn->next;
}
nc--;
}
```

```
int main()
{
single sl;
int no,pos,choice;
cout<<"Create Linked List";
char ch;
do
{
cout<<"\nEnter no: ";
cin>>no;
sl.create(no);
cout<<"do u want to continue(y/n)";
cin>>ch;
}while(ch=='y');
sl.display();
do
{ cout<<"\n**********";
cout<<"\n1.Insert\n2.Delete\n3.display\n4.exit ";
cout<<"\n enter choice: ";
cin>>choice;
cout<<"\n**********";
switch(choice)
{
case 1:
cout<<"\nEnter Position & no";
cin>>pos>>no;
sl.insertpos(pos,no);
break;
case 2:
cout<<"\nEnter Position";
cin>>pos;
sl.deletepos(pos);
break;
case 3:
sl.display();
```

```
break;

case 5:sl.sort();

break;

case 4: exit(0);

default:cout<<"\n wrong choice";

}

}while(choice<6);

}
```

4. Write a program for Sequential search.

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
 int a[30],n,s,count=0;
 cout<<"Enter the size of N:";
 cin>>n;
 cout<<"Enter "<<n<<" elements:";
 for(int i=0;i<n;i++)
 cin>>a[i];
 cout<<"Enter the search element:";
 cin>>s;
 for(int i=0;i<n;i++)
   if(a[i]==s){
              cout<<"Search Element found at location:"<<i;
              count++;
            }
            if(count==0)
            cout<<"Search Element not found";
 }
```

5. Write a program for Binary Search Technique

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
 int a[30],n,i,s,first,last,mid,count=0,temp;
 cout<<"Enter the size of N:";
 cin>>n;
 cout<<"Enter "<<n<<" elements:";
 for(i=0;i<n;i++)
 cin>>a[i];
 for(int j=0;j<n;j++)
 for(int k=0;k<n-1;k++)
 if(a[k]>=a[k+1])
 {
 temp=a[k];
 a[k]=a[k+1];
 a[k+1]=temp;
```

```
 }
 cout<<"Sorted Elements"<<endl;
 for(i=0;i<n;i++)
 cout<<a[i]<<endl;
cout<<"Enter the search element:";
 cin>>s;
 first = 0;
 last = n-1;
while(first <= last)
 {
 mid = (first+last)/2;
 if( a[mid] == s)
 {
cout<<"The element"<<s<<" is present at position "<<mid+1<<"in list";
count =1;
break;
 }
 else if(a[mid] < s)
first = mid+1;
 else
last = mid-1;
 }
 if( count == 0)
 cout<<"The element "<<s<<" is not present in the list";
}
```

6. Write a program to find Factorial of +ve Integer using Recursion.

```
#include<iostream>
#include<conio.h>
using namespace std;
int fact(int);
int main()
{
int n,f;
cout<<"Enter the value of n:";
cin>>n;
f=fact(n);
cout<<"Fact="<<f;
getch();
}
int fact(int n)
{
if(n>=1)
return n*fact(n-1);
else
return 1;
}
```

7. Write a program to find nth term of the Fibonacci sequence using Recursion

```
#include<iostream>
#include<conio.h>
using namespace std;

int fibo(int);
int main()
{
```

```
int num;
int result;
cout<<"Enter the value of n:";
cin>>num;
if(num<0)
cout<<"Fibonacci of negative number is not possible"<<endl;
else
{
result = fibo(num);
cout<<"The nth number in fibonacci series is"<<result;
getch();
}
}
int fibo(int num)
{
if (num == 0)
{
return 0;
}
else if (num == 1)
{
return 1;
}
else
{
return(fibo(num - 1) + fibo(num - 2));
}
}
```

8.Write a program to find GCD of two +ve integers using Recursion.

```
#include <iostream>
#include<conio.h>
using namespace std;

int gcd(int n1, int n2);
int main()
{
int n1,n2;
cout<<"Enter two positive integers: ";
cin>>n1>>n2;
int res=gcd(n1,n2);
cout<<"G.C.D of"<<n1<<"and "<<n2<<"is "<<res;
getch();
}
int gcd(int n1, int n2)
{
if (n2 != 0)
return gcd(n2, n1%n2);

else return n1;
}
```

9. Write a program for sorting the given list of numbers in ascending order using Bubble sort.

Bubble Sort:

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

```
using namespace std;
int main()
{
 int a[30],n,i, temp;
 cout<<"Enter the size of N:";
 cin>>n;
 cout<<"Enter "<<n<<" elements:";
 for(i=0;i<n;i++)
 cin>>a[i];
 for(int j=0;j<n;j++)
 for(int k=0;k<n-1;k++)
 if(a[k]>=a[k+1])
 {
 temp=a[k];
 a[k]=a[k+1];
 a[k+1]=temp;
 }
 cout<<"Bubble Sorted Elements"<<endl;
 for(i=0;i<n;i++)
 cout<<a[i]<<endl;
 getch();
}
```

10. Write a program for sorting the given list of numbers in ascending order using Selection sort.

Selection sort

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
 int a[30],n,i, temp,min;
 cout<<"Enter the size of N:";
 cin>>n;
 cout<<"Enter "<<n<<" elements:";
 for(i=0;i<n;i++)
 cin>>a[i];
for(int j=0;j<n-1;j++)
{
 min=j;  // min=0; a[min]
 for(int k=j+1;k<n;k++)
 if(a[k]<a[min])
   {
      min=k;
    temp=a[j];
   a[j]=a[min];
    a[min]=temp;}
}
 cout<<"Selection Sorted Elements"<<endl;
 for(i=0;i<n;i++)
 cout<<a[i]<<endl;
 getch();
}
```