1. (a). Write a program to implement the following using an array : **Stack ADT**

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

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;
}


void main()
{
int ch,val;
Stack S;
clrscr();
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);
}
}
}
```

1. (b). Write a program to implement the following using an array : **Queue ADT**

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

class Queue
{
 private:
              int Queue[10];
              int rear,front;
              int maxcapacity;
 public:
          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;
}
}


void main()
{
int ch,val;
Queue q;
clrscr();
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);
}
}
}
```

2. Write a program to convert the given infix expression to postfixexpression using stack.

```
#include<iostream.h>

#include<conio.h>


class Stack

{

private:

int Stack[10];

int MaxCapacity;

int top;


public:

Stack()

{

MaxCapacity = 10;

top = -1;

}

void push(char Element);

char pop();

char getTop();

int isEmpty();

};


void Stack :: push(char Element)
```

```
{
if(top==MaxCapacity-1)
cout<<"Stack overflow";
else
Stack[++top] = Element;
}

char Stack :: pop()
{
char ch;
if(top==-1)
cout<<"Stack Underflow";
else
return(Stack[top--]);
}

char Stack::getTop()
{
if(top==-1)
cout<<"Stack Underflow";
else
return(Stack[top]);
}
int Stack::isEmpty()
```

```
{
if(top==-1)
return 1;
else
return 0;
}


char isp(char ch)
{
switch(ch)
{
case '+':
case '-': return 1;
case '*':
case '/': return 2;
case '#': return -1;
}
}

char icp(char ch)
{
switch(ch)
{
case '+':
```

```
case '-': return 1;
case '*':
case '/': return 2;
}
}

void intopost(char infix[20])
{
int i=0;
char ch,x;
Stack s;
s.push('#');
while(infix[i]!='\0')
{
ch=infix[i];
i++;
if(ch>='a' && ch<='z')
cout<<ch;
else
{
while(isp(s.getTop())>=icp(ch))
{
x=s.pop();
cout<<x;
}
```

```
s.push(ch);
}
}
while(!s.isEmpty())
{
x=s.pop();
if(x!='#')
cout<<x;
}
}

void main()
{
char infix[20];
clrscr();
cout<<"Enter the infix expression";
cin>>infix;
cout<<"Postfix expression is: ";
intopost(infix);
}
```

3. Write a program to evaluate a postfix expression using stack.

```
#include<iostream.h>
#include<conio.h>
#include<ctype.h>
class Stack
{
private:
int Stack[10];
int MaxCapacity;
int top;

public:
Stack()
{
MaxCapacity = 10;
top = -1;
}
void push(int Element);
char pop();
int getTop();
int isEmpty();
};

void Stack :: push(int Element)
{
if(top==MaxCapacity-1)
cout<<"Stack overflow";
else
Stack[++top] = Element;
}

char Stack :: pop()
{
if(top==-1)
cout<<"Stack Underflow";
else
return(Stack[top--]);
}


int Stack::getTop()
{
if(top==-1)
cout<<"Stack Underflow";
else
return(Stack[top]);
}

int eval(char postfix[10])
```

```
{
int i=0;
char ch;
int n1,n2,n3;
Stack s;
while(postfix[i]!='\0')
{
ch=postfix[i];
if(isdigit(ch))
{
s.push(ch-'0');
}
else
{
n1=s.getTop();
s.pop();
n2=s.getTop();
s.pop();
switch(ch)
{
case '+': n3=n2+n1;
          break;
case '-': n3=n2-n1;
          break;
case '*': n3=n2*n1;
          break;
case '/': n3=n2/n1;
          break;
default: n3=0;
}
s.push(n3);
}
i++;
}
return(n3);
}

void main()
{
char postfix[10];
clrscr();
cout<<"Enter the Postfix expression";
cin>>postfix;
cout<<"Evaluated Postfix expression is: ";
int res=eval(postfix);
cout<<res;
getch();
}
```

4. Write a program to ensure the parentheses are nested correctly in an arithmetic expression.

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

class Stack
{
private:
int Stack[10];
int MaxCapacity;
int top;

public:
Stack()
{
MaxCapacity = 10;
top = -1;
}
void push(char Element);
char pop();
char getTop();
int isEmpty();
};

void Stack :: push(char Element)
{
if(top==MaxCapacity-1)
cout<<"Stack overflow";
else
Stack[++top] = Element;
}

char Stack :: pop()
{
char ch;
if(top==-1)
cout<<"Stack Underflow";
else
return(Stack[top--]);
}


char Stack::getTop()
{
if(top==-1)
cout<<"Stack Underflow";
else
return(Stack[top]);
}
int Stack::isEmpty()
```

```
{
if(top==-1)
return 1;
else
return 0;
}


char isp(char ch)
{
switch(ch)
{
case '+':
case '-': return 1;
case '*':
case '/': return 2;
case '#': return -1;
}
}

char icp(char ch)
{
switch(ch)
{
case '+':
case '-': return 1;
case '*':
case '/': return 2;
}
}

int CheckParen(char infix[20])
{
int i=0;
char ch,x;
Stack s;
s.push('#');
int nesting=0;
while(infix[i]!='\0')
{
ch=infix[i];
i++;

if(ch=='(' || ch==')')
{
switch (ch)
                {
                   case '(':
                     nesting++;
                     break;
                   case ')':
                     nesting--;
```

```
                    break;
                }
}
else
{
while(isp(s.getTop())>=icp(ch))
{
x=s.pop();
}

s.push(ch);
}
}
return nesting;
}
void main()
{
char infix[20];
clrscr();
cout<<"Enter the infix expression";
cin>>infix;
int res=CheckParen(infix);
if(res==0)
cout<<"Expression is Valid";
else
cout<<"Expression is Invalid";
getch();
}
```

5. (a) Write a program to find Factorial of +ve Integer using Recursion.

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

int fact(int);
void main()
{
int n,f;
clrscr();
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;
}
```

5. (b). Write a program to find n^{th} term of the Fibonacci Sequence using Recursion

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

int fibo(int);

void main()
{
int num;
int result;
clrscr();
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));
}
}
```

5. (c). Write a program to find GCD of two +ve integers using Recursion.

```
#include <iostream.h>
#include<conio.h>
int gcd(int n1, int n2);
void main()
{
  int n1, n2;
  clrscr();
  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;
}
```

6 Write a program to create a single linked list and write functions to implement the following operations.
a) Insert an element at a specified position
b) Delete a specified element in the list
c) Search for an element and find its position in the list
d) Sort the elements in the list ascending order

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
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 searchval(int);
	void sort();
};

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";
}

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--;

}
void single::searchval(int no)
{
int i=1,f;
       for(tn=root;tn!=NULL;tn=tn->next,i++)
       {
```

```
            if(tn->data==no)
            {
                  cout<<tn->data<<"is found in "<<i<<" position";
                  f=1;
                  break;
            }
      }
      if(f==0)
            cout<<"element is not found";
}

void single::sort()
{
int i,j,temp;
for(i=1;i<=nc;i++)
{
t1=root;
t2=t1->next;
for(j=i+1;j<=nc;j++)
{
if(t1->data > t2->data)
{
temp=t1->data;
t1->data=t2->data;
t2->data=temp;
}
t1=t1->next;
t2=t2->next;
}}
cout<<endl<<"after sorting \n ";
for(tn=root;tn!=NULL;tn=tn->next)
{
cout<<tn->data<<"->";
}
cout<<"NULL";
}

void main()
{
single sl;
int no,pos,choice;
clrscr();
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.search\n4.display\n5.sort\n6.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:
				cout<<"Enter element to search";
				cin>>no;
				sl.searchval(no);
				break;
			case 4:
				sl.display();
				break;
			case 5:sl.sort();
				break;
			case 6: exit(0);
			default:cout<<"\n wrong choice";
		}

	}while(choice<6);
}
```

8. Write a program to create singular circular linked lists and function to implement the following operations.
a) Insert an element at a specified position
b) Delete a specified element in the list
c) Search for an element and find its position in the list

```
//Program for Circular Linked List

#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
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 create(int);
	void insertpos(int,int);
	void deletepos(int);
	void display();
	void searchval(int);
	void sort();
};

void single::create(int no)
{
nn=new node(no);
nn->next=NULL;
nc++;
	if(root==NULL)
	{
```

```
        root=tail=nn;
        tail->next=root; //Modify 1
        }
        else
        {
                tail->next=nn;
                nn->next=root; //Modify 2
                tail=nn;
        }
}

void single::display()
{
        cout<<"\n number of nodes are: "<<nc;
        cout<<endl<<"Circular Linked List\n";   // Modify 3
        for(tn=root;tn->next!=root;tn=tn->next)   // Modify 4
                cout<<tn->data<<"->";
                cout<<tail->data <<"->(First Element)"<<root->data;       //
Modify 5
}


void single::insertpos(int p,int no)
{
int i;
nn=new node(no);
nn->next=NULL;

        if(p==1)
        {
                nn->next=root;
                root=nn;
                tail->next=root;   // Modify 6
        }
        else
          if(p>nc)
            {
             tail->next=nn;
             tail=nn;
              tail->next=root;   //Modify 7
             }
           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;
              tail->next=root; // Modify 8
         }
        else
         if(p==nc)
          {
            rn=tail;
            tn=root;
            while(tn->next!=tail)
              {
                tn=tn->next;
              }
            tn->next=root;   // Modify 9
            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--;

}
```

```
void single::searchval(int no)
{
int i=1,f;
	for(tn=root;tn!=NULL;tn=tn->next,i++)
	{
		if(tn->data==no)
		{
			cout<<tn->data<<"is found in "<<i<<" position";
			f=1;
			break;
		}
	}
	if(f==0)
		cout<<"element is not found";
}

void single::sort()
{
int i,j,temp;
for(i=1;i<=nc;i++)
{
t1=root;
t2=t1->next;
for(j=i+1;j<=nc;j++)
{
if(t1->data > t2->data)
{
temp=t1->data;
t1->data=t2->data;
t2->data=temp;
}
t1=t1->next;
t2=t2->next;
}}
cout<<endl<<"after sorting \n ";
for(tn=root;tn->next!=root;tn=tn->next)   // Modify 10
{
cout<<tn->data<<"->";
}
cout<<"NULL";
}

void main()
{
single sl;
int no,pos,choice;
clrscr();
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.search\n4.display\n5.sort\n6.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:
				cout<<"Enter element to search";
				cin>>no;
				sl.searchval(no);
				break;
			case 4:
				sl.display();
				break;
			case 5:sl.sort();
				break;
			case 6: exit(0);
			default:cout<<"\n wrong choice";
		}

	}while(choice<6);
}
```

9. (a) Write programs to implement the following using a single linked list using Stack ADT

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
class node
        {
        public:
                int data;
                node *next;
                node(int d)
                 {
                   data=d;
                 }
        };
class stack
{

        node *top,*nn,*tn;
public:
        stack()
        {
        top=NULL;
        }

        void push(int);
        void pop();
        void display();

};

void stack::push(int no)
{
nn=new node(no);
nn->next=NULL;

        if(top==NULL)
           top=nn;
        else
        {
                nn->next=top;
                top=nn;
        }
        cout<<"Element "<<no<<" is added";
}

void stack::display()
{

        if(top==NULL)
          cout<<"Empty Stack";
        else
        {
        cout<<"********************";
        cout<<endl<<"stack elements are\n";
        for(tn=top;tn!=NULL;tn=tn->next)
                cout<<endl<<tn->data;
        cout<<endl<<"********************";
        }
}
```

```
void stack::pop()
{
          if(top==NULL)
            cout<<"stack under flow";
          else
           {
             cout<<"\n deleted element is "<<top->data;
                    tn=top;
                    top=tn->next;
           }
}

void main()
{
stack s;
int no,choice;
clrscr();
          do
          {
                    cout<<"\n1.push \n2.pop\n3.display\n4.Exit";
                    cout<<"\nenter choice:";
                    cin>>choice;
                    switch(choice)
                    {
                              case 1:
                                        cout<<"\nEnter a number to Push:";
                                        cin>>no;
                                        s.push(no);
                                        break;
                              case 2:
                                        s.pop();
                                        break;
                              case 3:
                                        s.display();
                                        break;
                              case 4: exit(0);
                              default:cout<<"\n wrong choice";
                    }

          }while(choice<4);
}
```

9. (b) Write programs to implement the following using a single linked list using Queue ADT

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
class node
          {
          public:
                    int data;
                    node *next;
                    node(int d)
                     {
                        data=d;
                     }
          };
class queue
{
```

```
        node *front,*rear,*tn,*nn;
public:
        queue()
        {
        front=NULL;
        rear=NULL;
        }

        void insertq(int);
        void deleteq();
        void display();

};

void queue::insertq(int no)
{
nn=new node(no);
nn->next=NULL;

        if(rear==NULL)
            front=rear=nn;
        else
        {
                rear->next=nn;
                rear=nn;
        }
        cout<<"Element "<<no<<" is added";
}

void queue::display()
{
        if(front==NULL)
           cout<<"queue is empty";
        else
        {
        cout<<"********************";
        cout<<endl<<"queue elements are\n";
        for(tn=front;tn!=NULL;tn=tn->next)
                cout<<tn->data<<"    ";
        cout<<"********************";
        }
}

void queue::deleteq()
{
         if(front==NULL)
           cout<<"queue is empty";
         else
         {
            cout<<"\n deleted element is "<<front->data;
                tn=front;
                front=tn->next;
         }
}

void main()
{
queue q;
int no,choice;
clrscr();
        do
```

```
        {
                cout<<"\n1.insert\n2.delete\n3.display";
                cout<<"\nenter choice";
                cin>>choice;
                switch(choice)
                {
                        case 1:
                                cout<<"\nEnter no";
                                cin>>no;
                                q.insertq(no);
                                break;
                        case 2:
                                q.deleteq();
                                break;
                        case 3:
                                q.display();
                                break;
                        case 4: exit(0);
                        default:cout<<"\n wrong choice";
                }

        }while(choice<4);
}
```

10 Write a program to implement Binary search technique

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int a[30],n,i,s,first,last,mid, count=0,temp;
  clrscr();
  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<<"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";
  getch();
}
```

11. Write a program for sorting the given list numbers in ascending order using the following technique: Bubble sort and Selection sort

Bubble Sort:

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

void main()
{
  int a[30],n,i, temp;
  clrscr();
  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();
}
```

Selection Sort:

```
#include<iostream.h>
#include<conio.h>
void main()
{
   int a[30],n,i, temp,min;
   clrscr();
   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;
  for(int k=j+1;k<n;k++)
     if(arr[k]<arr[min])
          min=k;
          temp=arr[j];
          arr[j]=arr[min];
               arr[min]=temp;
}

   cout<<"Selection Sorted Elements"<<endl;
   for(i=0;i<n;i++)
   cout<<a[i]<<endl;
   getch();
}
```

12. Write a program for sorting the given list numbers in ascending order using the following technique: Insertion sort and Quick sort

Insertion Sort:

```
#include<iostream.h>
#include<conio.h>
void main()
{
   int arr[30],n,i,j,k temp,min;
   clrscr();
   cout<<"Enter the size of N:";
   cin>>n;
   cout<<"Enter "<<n<<" elements:";
   for(i=0;i<n;i++)
   cin>>arr[i];

 for(int j=1;j<n;j++)
```

```
{
temp=arr[j];
k=j;
while(k>0 && arr[k-1]>=temp)
{
arr[k]=arr[k-1];
--k;
}
arr[k]=temp;
}

  cout<<"Insertion Sorted Elements"<<endl;
  for(i=0;i<n;i++)
  cout<<arr[i]<<endl;
  getch();
}
```

QuickSort:

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

 void QSort(int a[],int first,int last)
 {
  int temp,pivot,i,j;
  if(first<last)
  {
   i=first;
   j=last;
   pivot=a[first];
   while(i<j)
   {
    while(a[i]<=pivot&&i<last)
    i++;
    while(a[j]>pivot)
    j--;
    if(i<j)
    {
     temp=a[i];
     a[i]=a[j];
     a[j]=temp;
    }
   }
   temp=a[first];
   a[first]=a[j];
   a[j]=temp;
   QSort(a,first,j-1);
   QSort(a,j+1,last);
```

```
 }
}
void main()
{

 int n,i,a[20];
 clrscr();
 cout<<"Enter Array Size";
 cin>>n;

 cout<<"Enter Array Elements:";
 for(i=0;i<n;i++)
 cin>>a[i];

 QSort(a,0,n-1);

 cout<<"After Sorting:";
 for(i=0;i<n;i++)
 cout<<a[i]<<" " ;
 getch();
}
```