# 1. MERGE TWO ARRAY INTO SINGLE ARRAY

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a[50],b[50],c[50],i,j,k,n,n1,n2;
clrscr();
cout<<"\nEnter the first array size:";
cin>>n1;
cout<<"\nEnter the array elements:";
for(i=0;i<n1;i++)
{
cin>>a[i];
}
cout<<"\nEnter the second array size:";
cin>>n2;
cout<<"\nEnter the array elements:";
for(i=0;i<n2;i++)
{
cin>>b[i];
}
for(i=0;i<n1;i++)
{
c[i]=a[i];
}
n=n1+n2;
for(i=0,k=n1;k<n&&i<n2;i++,k++)
{
c[k]=b[i];
}
cout<<"\nArrays are after merge is:";
for(i=0;i<n;i++)
{
cout<<c[i]<<" ";
}
getch();
}
```

## 1. MERGE TWO ARRAY INTO SINGLE ARRAY

**INPUT/OUTPUT:**

Enter the first array size: 5

Enter the array elements: 1
2
3
4
5

Enter the second array size: 5

Enter the array elements: 6
7
8
9
10

Arrays are after merge is: 1 2 3 4 5 6 7 8 9 10

## 2. ARRAY IMPLEMENTATION USING STACK

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

```
#include<conio.h>
#include<stdlib.h>
#define MAX_SIZE 5
int main()
{
  clrscr();
  int item,choice,i;
  int arr_stack[MAX_SIZE];
  int top=0;
  int exit=1;
  cout<<" \n single stack examples ";
  do
  {
   cout<<"\n\n stack main menu ";
   cout<<"\n 1. push\n 2.pop\n3.display \n4.exit\n";
   cout<<"Enter your choice: ";
   cin>>choice;
   switch(choice)
   {
    case: 1
    if(top==MAX_SIZE)
    cout<<"\n stack is full";
    else
    {
    cout<<"\n Enter the value to be pushed: ";
    cin>>item;
    cout<<"\n position: "<<top<<"pushed value: "<<item;
     arr_stack[top++]=item;
     }
     break;
     case: 2
     if(top==0)
     cout<<"\n stack is empty";
     else
     {
      -- top;
      cout<<"\n position: "<<top<<"popped value: "<<arr_stack[top];
      }
      break;
      case: 3
      cout<<"\n stack size: "<<top;
      for(i=(top-1);i>=0;i--)
      cout<<"\n position: "<<i<<"value: "<<arr_stack[i];
      break;
      default:
      exit=0;
      break;
     }
   }while(exit);
   return 0;
}
```

**INPUT/OUTPUT:**

Single stack example

Stack main menu

1.push

2. pop
3. display
4. exit
Enter your choice: 1
Enter the value to be pushed: 5
Position:0 pushed value: 5
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 1
Enter the value to be pushed: 8
Position:1 pushed value: 8
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 1
Enter the value to be pushed: 3
Position:2 pushed value: 3
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 1
Enter the value to be pushed: 9
Position:3 pushed value: 9
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 1
Enter the value to be pushed: 6
Position:4 pushed value: 6
Enter your choice: 1
Stack is full

Stack main menu
1.push
2. pop
3. display
4. exit

Enter your choice: 3
Stack size: 5
Position: 4 value: 6
Position: 3 value: 9
Position: 2 value: 3
Position: 1 value: 8
Position: 0 value: 5
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 2
Position: 4 popped value: 6
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 2
Position: 3 popped value: 9
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 2
Position: 2 popped value: 3
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 2
Position: 1 popped value: 8
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 2
Position: 0 popped value: 5
Stack main menu
1.push
2. pop

3. display
4. exit
Enter your choice:2
Stack is empty
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 3
Stack size: 0
Stack main menu
1.push
2. pop
3. display
4. exit
Enter your choice: 4

## 3. ARRAY IMPLEMENTATION OF QUEUE

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#define  MAX_SIZE 100
Void main()
{
```

```
  Clrscr();
  int item,choice,i ;
  int arr_queue[MAX_SIZE];
  int rear=0;
  int front=0;
  int exit=1;
  cout<<"\n\t simple queue";
  do
  {
     Cout<<"\n\n\t  queue";
     Cout<<"\n\t1.insert\n\t2.remove\n\t3.display\n\t4.exit";
     Cout<<"\n\t enter your choice: ";
     Cin>>choice;
     Switch(choice)
     {
      Case :1
      if(rear==MAX_SIZE)
      cout<<"\n\t queue reached max!";
      else
      {
       Cout<<"\n\t enter the value to be insert: ";
       Cin>>item;
       Cout<<"\n\t position: "<<rear+1<<"insert value:"<<item;
       arr_queue[rear++]=item;
       }
       break;
       case: 2
       if(front==rear)
       cout<<"\n\t queue is empty";
       else
       {
        Cout<<"\n\t position: "<<front<<"remove value: "<<arr_queue[front];
        front++;
        }
        break;
        case: 3
        cout<<"\n\t queue size: "<<(rear-front);
        for(i=front;i<rear;i++)
        cout<<"\n\t position: "<<i<<"value: "<<arr_queue[i];
        break;
        default:
            exit=0;
          }
         }
         While(exit);
}
```

## 3. ARRAY IMPLEMENTATION OF QUEUE

**INPUT/OUTPUT:**

Simple queue
queue
1.insert
2. remove
3. display

4. exit
Enter your choice: 1
Enter the value to be insert: 6
Position:1 insert value: 6
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 1
Enter the value to be insert: 8
Position:2 insert value: 8
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 1
Enter the value to be insert : 4
Position:3 insert value: 4
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 1
Enter the value to be insert: 2
Position:4 insert value: 2
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 1
Enter the value to be insert: 9
Position:5 insert value: 9
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 3
queue size: 5
Position: 0 value: 6
Position: 1 value: 8

Position: 2 value: 4
Position: 3 value: 2
Position: 4 value: 9
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 2
Position: 0 remove value: 6
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 2
Position: 1 remove value: 8
queue
1.insert
2. remove
3. display
4. exit

Enter your choice: 2
Position: 2 remove value: 4
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 2
Position: 3 remove value: 2
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 2
Position: 2  remove value: 9
queue
1.insert
2. remove
3. display
4. exit
Enter your choice:2
queue is empty

```
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 3
queue size: 0
queue
1.insert
2. remove
3. display
4. exit
Enter your choice: 4
```

## 4. EVALUATION OF POSTFIX

```
#include<iostream.h>
#include<conio.h>
#include<ctype.h>
int top=-1;
int stk[40];
void push (int x)
{
top++;
stk[top]=x;
}
```

```
int pop(int y)
{
int x;
x=stk[y];
top--;
return x;
}
void main()
{
clrscr();
char arr[40];
cout<<"\nEnter postfix expression:";
cin>>arr;
char*p;
p=arr;
int temp;
while(*p!='\0')
{
if(isalnum(*p))
push(*p-48);
else if(*p=='+')
{
temp=pop(top)+pop(top);
push(temp);
}
else if(*p=='-')
{
temp=pop(top-1)-pop(top+1);
push(temp);
}
else if(*p=='*')
{
temp=pop(top)*pop(top);
push(temp);
}
else if(*p=='/')
{
temp=pop(top-1)/pop(top+1);
push(temp);
}
p++;
}
while(top!=-1)
{
cout<<"\nPostfix evaluted result:"<<stk[top];
top--;
}
getch();
```

}

**INPUT/OUTPUT:**

Enter postfix expression: 234*6/+
Postfix evaluated result: 4

## 5.SINGLY LINKED LIST

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int info;
struct node*next;
}*start;
class sll
{
public:
      node*create(int);
      void add();
      void delet();
      void display();
```

```
sll()
{
start=NULL;
}
};
node*sll::create(int v)
{
struct node*s,*temp;
temp=new(struct node);
if(temp==NULL)
{
cout<<"\nMemory not allocated"<<endl;
return 0;
}
else
{
temp->info=v;
temp->next=NULL;
return temp;
}
}
void sll::add()
{
int v;
cout<<"\nEnter the value:";
cin>>v;
struct node*temp,*p;
temp=create(v);
if(start==NULL)
{
start=temp;
start->next=NULL;
}
else
{
p=start;
start=temp;
start->next=p;
}
cout<<"\nElement inserted"<<endl;
}
void sll::delet()
{
struct node*temp;
temp=start;
start=start->next;
cout<<"\nElement deleted"<<endl;
free(temp);
return;
}
void sll::display()
{
```

```
struct node*q;
if(start==NULL)
{
cout<<"\nLists empty"<<endl;
return;
}
q=start;
cout<<"\nThe singly linked list is:"<<endl;
while(q!=NULL)
{
cout<<q->info<<"<->";
q=q->next;
}
cout<<"NULL"<<endl;
}
void main()
{
int c,e,p;
sll s;
clrscr();
while(1)
{
cout<<"\nOperations on singly linked list";
cout<<"\n1.INSERT \n2.DELETE \n3.DISPLAY \n4.EXIT";
cin>>c;
switch(c)
{
case 1:
        s.add();
        break;
case 2:
        if(start==NULL)
        {
        cout<<"list empty";
        break;
        }
        s.delet();
        break;
case 3:
        s.display();
        break;
case 4:
        exit(1);
default:
        cout<<"Wrong choice";
        }
        }
        getch();
        }
```

INPUT/OUTPUT:

Operations on Singly linked list
1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
1
Enter the value: 10
Element inserted

Operations on Singly linked list

1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
1
Enter the value: 20
Element inserted

Operations on Singly linked list

1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
1
Enter the value: 30
Element inserted
Operations on Singly Linked List
1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
3
The singly linked list
30<->20<->10<->NULL

Operations on Singly Linked List

1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
2
Element deleted

Operations on Singly Linked List

1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
3
The singly linked list
20<->10<->NULL

Operations on Singly Linked List
1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
4.

## 6. POLYNOMIAL ADDITION

```
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
struct poly
{
int coeff;
int pow;
poly *next;
};
class add2poly
{
poly *poly1,*poly2,*poly3;
public:
add2poly()
{
poly1=poly2=poly3=NULL;
}
void addpoly();
void display();
};
```

```
void add2poly::addpoly()
{
int i,p;
poly *new1=NULL,*end=NULL;
cout<<"\nEnter highest power of x:";
cin>>p;
cout<<"\nFirst polynomial";
for(i=p;i>=0;i--)
{
new1=new poly;
new1->pow=p;
cout<<"\nEnter coeffecient for degree"<<i<<"::";
cin>>new1->coeff;
new1->next=NULL;
if(poly1==NULL)
poly1=new1;
else
end->next=new1;
end=new1;
}
cout<<"\nSecond polynomial";
end=NULL;
for(i=p;i>=0;i--)
{
new1=new poly;
new1->pow=p;
cout<<"\nEnter coefficient for degree"<<i<<"::";
cin>>new1->coeff;
new1->next=NULL;
if(poly2==NULL)
poly2=new1;
else
end->next=new1;
end=new1;
}
poly *p1=poly1,*p2=poly2;
end=NULL;
while(p1!=NULL && p2!=NULL)
{
if(p1->pow==p2->pow)
{
new1=new poly;
new1->pow=p;
new1->pow=p--;
new1->coeff=p1->coeff+p2->coeff;
new1->next=NULL;
if(poly3==NULL)
poly3=new1;
else
end->next=new1;
end=new1;
}
```

```
p1=p1->next;
p2=p2->next;
}
}
void add2poly::display()
{
poly *t=poly3;
cout<<"\nAnswer after additionis:";
while(t!=NULL)
{
cout.setf(ios::showpos);
cout<<t->coeff;
cout.unsetf(ios::showpos);
cout<<"x"<<t->pow;
t=t->next;
}
}
void main()
{
clrscr();
add2poly obj;
obj.addpoly();
obj.display();
getch();
}
```

## 6. POLYNOMIAL ADDITION

**INPUT/OUTPUT:**

Enter highest power of x: 4

FIRST POLYNOMIAL

Enter coefficient for degree: 4::2
3::3
2::4
1::5
0::6

SECOND POLYNOMIAL

Enter coefficient for degree:4::2
3::3
2::4
1::5
0::6

Answer after addition is:+4x4+6x3+8x2+10x1+12x0

## 7.DOUBLY LINKED LIST

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *next;
struct node *prev;
}*start;
class dll
{
public:
      void create(int v);
      void add(int v);
      void del();
      void disp();
      dll()
      {
      start=NULL;
      }
      };
      void dll::create(int v)
      {
```

```
struct node *s,*temp;
temp=new(struct node);
temp->info=v;
temp->next=NULL;
if(start==NULL)
{
temp->prev=NULL;
start=temp;
}
else
{
s=start;
while(s->next!=NULL)
s=s->next;
s->next=temp;
temp->prev=s;
}
};
void dll::add(int v)
{
if(start==NULL)
{
cout<<"\nFirst create the list:"<<endl;
}
struct node *temp;
temp=new(struct node);
temp->prev=NULL;
temp->info=v;
temp->next=start;
start->prev=temp;
start=temp;
cout<<"\nElement inserted"<<endl;
}
void dll::del()
{
struct node *temp;
//if(start->info==v)
//{
temp=start;
start=start->next;
start->prev=NULL;
cout<<"\nElement deleted"<<endl;
free(temp);
}
void dll::disp()
{
struct node *q;
if(start==NULL)
```

```
{
cout<<"list empty"<<endl;
return;
}
q=start;
cout<<"The doubly linked list is:"<<endl;
while(q!=NULL)
{
cout<<q->info<<"<->";
q=q->next;
}
cout<<"NULL"<<endl;
}
void main()
{
int c,e;
clrscr();
dll d;
while(1)
{
cout<<"\nOperations on doubly linked list";
cout<<"\n1.CREATE NODE \n2.INSERT \n3.DELETE \n4.DISPLAY \n5.EXIT";
cout<<"\nEnter your choice:";
cin>>c;
switch(c)
{
case 1:
        cout<<"\nEnter the element:";
        cin>>c;
        d.create(c);
        break;
case 2:
        cout<<"\nEnter the element:";
        cin>>c;
        d.add(c);
        break;
case 3:
        if(start==NULL)
        {
        cout<<"list empty";
        break;
        }
        cin>>c;
        d.del();
        break;
case 4:
        d.disp();
        break;
```

```
case 5:
        exit(1);
default:
        cout<<"Wrong choice";
        }
        }
       //getch();
        }
```

## 7.DOUBLY LINKED LIST

INPUT/OUTPUT:

```
Operations on Doubly linked list
1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
1
Enter the value: 10
Element inserted

Operations on Doubly linked list

1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
1
Enter the value: 20
Element inserted

Operations on Doubly linked list

1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
1
Enter the value: 30
Element inserted
Operations on Doubly Linked List
```

1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
3
The doubly linked list
30<->20<->10<->NULL

Operations on Doubly Linked List

1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
2
Element deleted

Operations on Doubly Linked List

1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
3
The doubly linked list
20<->10<->NULL

Operations on Doubly Linked List
1.INSERT
2.DELETE
3.DISPLAY
4.EXIT
4

## 8. BINARY TREE TRAVERSAL

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
struct treenode
{
int data;
treenode *left;
treenode *right;
};
treenode *insert(treenode *node)
{
if(node==NULL)
{
node=new treenode;
cout<<"\nEnter the number:";
cin>>node->data;
node->left=node->right=NULL;
return node;
}
char c;
cout<<"\nEnter the choice:";
cin>>c;
if(c=='r'|| c=='R')
{
node->right=insert(node->right);
}
if(c=='l'|| c=='L')
{
node->left=insert(node->left);
```

```
}
return node;
}
void inorder(treenode *node)
{
if(node==NULL)
{
return;
}
inorder(node->left);
cout<<node->data<<" ";
inorder(node->right);
}
void preorder(treenode *node)
{
if(node==NULL)
{
return;
}
cout<<node->data<<" ";
preorder(node->left);
preorder(node->right);
}
void postorder(treenode *node)
{
if(node==NULL)
{
return;
}
postorder(node->left);
postorder(node->right);
cout<<node->data<<" ";
}
void main()
{
treenode *root=NULL,*temp;
char ch;
clrscr();
while(1)
{
root=insert(root);
cout<<"\nDo you want to continue:Y or N:";
cin>>ch;
if(ch=='n'||ch=='N')
break;
}
cout<<endl<<"Inorder is:";
inorder(root);
```

```
cout<<endl<<"Preorder is:";
preorder(root);
cout<<endl<<"Postorder is:";
postorder(root);
getch();
}
```

## 8. BINARY TREE TRAVERSAL

**INPUT/OUTPUT:**

```
Enter the number: 30

Do you want to continue:Y or N:
Y

Enter the choice:
L

Enter the number: 10

Do you want to continue:Y or N:
Y

Enter the choice:
R

Enter the number: 40

Do you want to continue:Y or N:
N

Inorder is: 10  30  40
Preorder is: 30  10  40
Postorder is: 10  40  30
```

# 9. BREATH FIRST SEARCH(BFS)

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
int cost[10][10],i,j,k,n,qu[10],front,rare,v,visited[10],visit[10];
void main()
{
int m;
clrscr();
cout<<"\nEnter number of vertices:";
cin>>n;
cout<<"\nEnter number of edges:";
cin>>m;
cout<<"\NEDGES:";
for(k=1;k<=m;k++)
{
cin>>i>>j;
cost[i][j]=1;
}
cout<<"\nEnter initial vertex:";
cin>>v;
cout<<"\nOrder of visited vertices:";
cout<<v<<" ";
visited[v]=1;
k=1;
while(k<n)
{
for(j=1;j<=n;j++)
{
if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
{
visit[j]=1;
qu[rare++]=j;
}
}
v=qu[front++];
cout<<v<<" ";
k++;
visit[v]=0; visited[v]=1;
}
getch();
}
```

**INPUT/OUTPUT:**

```
Enter number of vertices: 4
Enter number of edges: 4
EDGES:
    1    2
    2    1  3
    3    2  4
    4    3  4
Enter initial vertex: 1
Ordered of visited vertices: 1 2 3 4
```

# 10. DEPTH FIRST SEARCH(DFS)

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
int cost[10][10],i,j,k,n,v,top,stk[10],visit[10],visited[10];
void main()
{
int m;
clrscr();
cout<<"\nEnter number of vertices:";
cin>>n;
cout<<"\nEnter number of edges:";
cin>>m;
cout<<"\nEDGES:";
for(k=1;k<=m;k++)
{
cin>>i>>j;
cost[i][j]=1;
}
cout<<"\nEnter initial vertex:";
cin>>v;
cout<<"\nOrder of visited vertices:";
cout<<v<<" ";
visited[v]=1;
k=1;
while(k<n)
{
for(j=1;j>=1;j--)
if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
{
visit[j]=1;
stk[top++]=j;
}
v=stk[--top];
cout<<v<<" ";
k++;
visit[v]=0; visited[v]=1;
}
getch();
}
```

**INPUT/OUTPUT:**

```
Enter  of vertices: 4
Enter of edges: 4
EDGES:
     1    2
     2    1  3
     3    2  4
     4    3  4
Enter initial vertex: 1
Ordered of visited vertices:  1  2  3  4
```

## 11. SEQUENTIAL SEARCH

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a[5],n,i,x,pos,c=0;
clrscr();
cout<<"\nEnter the array size:";
cin>>n;
cout<<"\nEnter the array elements:";
for(i=0;i<n;i++)
{
cin>>a[i];
}
cout<<"\nEnter the search element:";
cin>>x;
for(i=0;i<n;i++)
{
if(a[i]==x)
{
c=1;
pos=i+1;
break;
}
else
c=0;
}
if(c==1)
cout<<"Search element"<<x<<"is present in array at position"<<pos;
else
cout<<"Search element is not present in given array";
getch();
}
```

**INPUT/OUTPUT:**

```
Enter the array size: 5
Enter the array elements:    1
                             2
                             3
                             4
                             5
Enter the search element: 3
Search element 3 is present in array at position 3

Enter the search element 6:
Search element is not present in given array
```

## 12. BINARY SEARCH

```
#include<iostream.h>
#include<conio.h>
void main()
{
int arr[10],n,i,first,middle,last,search;
clrscr();
cout<<"\nEnter total number of elements";
cin>>n;
cout<<"\nEnter"<<n<<"number:";
for(i=0;i<n;i++)
{
cin>>arr[i];
}
cout<<"\nEnter a number to find:";
cin>>search;
first=0;
last=n-1;
middle=(first+last)/2;
while(first<=last)
{
if(arr[middle]<search)
{
first=middle+1;
}
else if(arr[middle]==search)
{
cout<<"Search element"<<"found at location"<<middle+1<<"\n";
break;
}
else
{
last=middle-1;
}
middle=(first+last)/2;
}
if(first>last)
{
cout<<"Not found!"<<search<<"is not present in the list";
}
getch();
}
```

## 12. BINARY SEARCH

**INPUT/OUTPUT:**

```
Enter total number of elements: 5
Enter 5 number :       1
                       2
                       3
                       4
                       5
Enter a number to find: 3
3 is found at location:   3

Enter a number to find: 6
Not found! Search element 6 is not present given in the list
```

# 13. QUICK SORT

```
#include<iostream.h>
#include<conio.h>
int part(int low,int high,int *a)
{
int i,p,t,h=high,l=low;
p=a[low];
while(low<high)
{
while(a[l]<p)
{
l++;
}
while(a[h]>p)
{
h--;
}
if(l<h)
{
t=a[l];
a[l]=a[h];
a[h]=t;
}
else
{
t=p;
p=a[l];
a[l]=t;
break;
}
}
return h;
}
void quick(int l,int h,int*a)
{
int index,i;
if(l<h)
{
index=part(l,h,a);
quick(l,index-l,a);
quick(h,index+l,a);
}
}
int main()
{
clrscr();
int a[100],n,l,h,i;
cout<<"\nEnter the number of elements:";
cin>>n;
cout<<"\nEnter the elements(use space as a separator):";
for(i=0;i<n;i++)
```

```
cin>>a[i];
cout<<"\nInitial array:";
for(i=0;i<n;i++)
{
cout<<a[i]<<"\t";
}
h=n-1;
l=0;
quick(l,h,a);
cout<<"\nAfter sorting:";
for(i=0;i<n;i++)
{
cout<<a[i]<<"\t";
}
getch();
return 0;
}
```

**INPUT/OUTPUT:**

Enter the number of elements: 5
Enter the elements (Use space as a separator): 5
4
3
2
1
Initial array: 5 4 3 2 1
After sorting: 1 2 3 4 5

# 14. SELECTION SORT

```
#include<iostream.h>
#include<conio.h>
template<class T>
void sort(T a[],int n)
{
int i,j,t;
clrscr();
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(a[j]<a[i])
{
t=a[i];
a[i]=a[j];
a[j]=t;
}
}
}
}
int main()
{
int a[100],i,n;
clrscr();
cout<<"\nEnter the number of element:";
cin>>n;
cout<<"\nEnter the elements:";
for(i=0;i<n;i++)
{
cout<<"\nENTER:";
cin>>a[i];
}
sort(a,n);
cout<<"\nAfter sorting:";
for(i=0;i<n;i++)
{
cout<<a[i]<<"\t";
}
getch();
return 0;
}
```

**INPUT/OUTPUT:**

```
Enter the number of elements: 5
Enter the elements:
ENTER: 3
ENTER: 1
ENTER: 5
ENTER: 2
ENTER: 4
After sorting:1  2  3  4  5
```

# 15. INSERTION SORT

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
int n,a[10],i,j,temp;
cout<<"\nEnter array size:";
cin>>n;
cout<<"\nEnter array elements:";
for(i=1;i<=n;i++)
{
cin>>a[i];
}
for(i=1;i<=n;i++)
{
temp=a[i];
j=i-1;
while((temp<a[j])&&(j>0))
{
a[j+1]=a[j];
j=j-1;
}
a[j+1]=temp;
}
cout<<"\nSorted list:";
for(i=1;i<=n;i++)
{
cout<<a[i]<<" ";
}
getch();
}
```

**INPUT/OUTPUT:**

```
Enter array size: 5
Enter array elements: 10
                      6
                      4
                      2
Sorted list: 2  4  6  8  10
```