EX.NO: 1

BINARY SEARCH

```
#include<iostream.h>
#include<conio.h>
void main()
{
	int i,k,n,a[20],f=0,last,first=0,middle;
	clrscr();
	cout<<"Enter the number of elements:";
	cin>>n;
	last=n-1;
	cout<<"Enter the elements one by one :";
	for(i=0;i<n;i++)
	cin>>a[i];
	cout<<"Enter the searching element :";
	cin>>k;
	while(first<=last)
	{
		middle=(first+last)/2;
		if(a[middle]>k)
		{
			last=middle-1;
		}
		else if(a[middle]<k)
		{
		first=middle+1;
		}
		else
		{
			f=1;
			break;
		}
	}
	if(f==1)
	cout<<"The element found at position "<<middle+1;
	getch();
}
```

OUTPUT:

BINARY SEARCH

Enter the number of elements :

5

Enter the elements by one :

7
4
6
8
9

Enter the searching element :

8

The element found at position 4

EX.NO : 2

STACK IMPLEMENTATION

.

```
#include<iostream.h>
#include<conio.h>
struct node
{
        int data;
        struct node*link;
};
typedef struct node n;
 n*head,*last,*p;
void main()
{
        int ch;
        void push();
        void pop();
        void display();
        clrscr();
        head=NULL;
        do
        {
                cout<<"\n1.push\n2.pop\n3.display\n4.exit\n";
                 cout<<"\n Enter the choice    ";
                cin>>ch;
                switch(ch)
                {
                        case 1: push();
                         break;
                        case 2: pop();
                        break;
                        case 3: display();
                        break;
                }
        }
while(ch!=4);
getch();
}
void push()
{
        p=new node;
        cout<<"\n Enter the data to be inserted:";
        cin>>p->data;
```

```
        if(head==NULL)
        p→link=NULL;
        else
        p→link=head;
        head=p;
}
void pop()
{
        if(head==NULL)
        cout<<"\n Stack is empty"; else
        {
                p=head;
                cout<<"\n Deleted data is:"<<p->data; head=head->link;
                delete(p);

        }
}
void display()
{
        if(head==NULL)
        cout<<"\n stack is Empty"; else
        {
                p=head;
                while(p!=NULL)
                {
                        cout<<p->data<<endl; p=p->link;
                }
        cout<<endl;
        }
}
```

OUTPUT:

STACK IMPLEMENTATION

```
1.push
2.pop
3.display
4.exit

Enter the choice    1
Enter the data to be inserted: 34

1.push
2.pop
3.display
4.exit

Enter the choice    1
Enter the data to be inserted: 23
1.push
2.pop
3.display
4.exit

Enter the choice    1
Enter the data to be inserted: 56
1.push
2.pop
3.display
4.exit

Enter the choice    3
56
23
34

1.push
2.pop
3.display
4.exit

Enter the choice    2
```

Deleted data is: 56

1.push
2.pop
3.display
4.exit

Enter the choice 2
Deleted data is: 23

1.push
2.pop
3.display
4.exit

Enter the choice 4

EX.NO : 3

QUEUE IMPLEMENTATION

```
#include<iostream.h>
#include<conio.h>
#define QUEUE_SIZE 100
int queue[QUEUE_SIZE];
void main()
{
        int noption,front=-1,rear=-1,i; clrscr();
        cout<<"\n\n\t\t**QUEUE IMPLEMENTATION**\t\t\n";
        cout<<"\n\n\t\t ---------------------------------------- \t\t\n\n";
        do
        {
                cout<<"\n1.Add\n2.Delete\n3.print\n4.Exit";
                cout<<"\n Enter the option:";
                cin>>noption;
                switch(noption)
                {
                        case 1:if(rear<QUEUE_SIZE)
                        {
                                cout<<"\n Enter the Data:";
                                cin>>queue[++rear];
                        }
                        else
                        cout<<"\n QUEUE full";
                        break;
                        case 2:if(front<rear)
                        cout<<"\n Data"<<queue[++front]<<endl;
                        else
                        cout<<"\n queue is Empty";
                        break;
                        case 3:
                        cout<<"\n QUEUE\t";
                        i=front+1;
                while(i<=rear)
                cout<<"\n"<<queue[i++]<<"\t";
                cout<<"\n";
                }
        }
        while(noption<=3);
}
```

OUTPUT:

****QUEUE IMPLEMENTATION****

1.Add
2.Delete
3.print
4.Exit

Enter the option: 1
Enter the Data: 34

1.Add
2.Delete
3.print
4.Exit

Enter the option: 1
Enter the Data: 35

1.Add
2.Delete
3.print
4.Exit

Enter the option: 1
Enter the Data: 67

1.Add
2.Delete
3.print
4.Exit

Enter the option: 3

QUEUE

34
35
67

1.Add

2.Delete
3.print
4.Exit

Enter the option: 2
Data 34

1.Add
2.Delete
3.print
4.Exit

Enter the option: 2
Data 35

1.Add
2.Delete
3.print
4.Exit

Enter the option:3

QUEUE 67

1.Add
2.Delete
3.print
4.Exit

Enter the option:4

EX.NO : 4

BINARY TREE TRAVERSAL

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class node
{
        public:
        int data;
        node*left;
        node*right;
};
class treetrav
{
        public:
        node*create(node*,int);
        void preodr(node*);
        void inodr(node*);
        void postodr(node*);
};
node*treetrav::create(node*root,int x)
{
        if(root==NULL)
        {
                root=new node;
                root->data=x;
                root->right=NULL;
                root->left=NULL;
                return (root);
        }
        else
        {
                if(x<root->data)
                root->left=create(root->left,x); else
                root->right=create(root->right,x); r}eturn(root);
        }
}
void treetrav::preodr(node*root)
{
        if(root!=NULL)
        {
        cout<<" "<<root->data;
```

```
        preodr(root->left);
        preodr(root->right);
        }
}
void treetrav::inodr(node*root)
{
        if(root!=NULL)
        {
                inodr(root->left);
                cout<<" "<<root->data;
                inodr(root->right);
        }
}
void treetrav::postodr(node*root)
{
        if(root!=NULL)
        {
                postodr(root->left);
                postodr(root->right);
                cout<<" "<<root->data;
        }
}
void main()
{
        node*root=NULL;
        treetrav t;
        int x;
        char cho;
        clrscr();
        cout<<"\n\n\t\t ****BINARY TREE TRAVERSAL*****\t\t\n\n";
        cout<<"\n\n Enter The Elements:";
        cin>>x;
        cho='Y';
        while(cho=='Y')
        {
                root=t.create(root,x);
                cout<<"\n Do You Want To Continue(Y/N)?";
                cin>>cho;
                if(cho=='N')
                break;
                else if(cho=='Y')
                {
                        cout<<"\n Enter The Element:";
                        cin>>x;
```

```
            }
                else
                goto L;
        }
        cout<<"\n\n\t\t Binary Tree Traversal";
        cout<<"\n\n\t\t====================";
        cout<<"\n\n\t\t preoder traversal";
        t.preodr(root);
        cout<<"\n\n\t\t inodr traversal";
        t.inodr(root);
        cout<<"\n\n\t\t postoder traversal";
        t.postodr(root);
        getch();
}
```

OUTPUT:

*********BINARY TREE TRAVERSAL**********

Enter The Elements: 4

Do You Want To Continue(Y/N)? Y
Enter The Element:5
Do You Want To Continue(Y/N)? Y
Enter The Element:2
Do You Want To Continue(Y/N)? N

Binary Tree Traversal

==========

preoder traversal 4 2 5
inodr traversal 2 4 5
postoder traversal 2 5 4

Ex.No :5

BREADTH FIRST SEARCH

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
int visited[10],n,a[10][10];
class que
{
        int a[10];
        int f,r;
        public:
        que()
        {
                f=r=0;
        }
        void enqueue(int);
        int dequeue();
};

void que::enqueue(int x)
{
        r=r+1;
        a[r]=x;
}
int que::dequeue()
{
        if(f!=r)
        {
                f=f+1;
                int x=a[f];
                return x;
        }
        else
        return-1;
}
void main()
{
        int i,j;
        void bfs(int v);
        clrscr();
        cout<<"\n\t\t***BREADTH FIRST SEARCH***\t\t\n";
        cout<<"\nEnter the number of vertices in the graph:";
```

```
	cin>>n;
	if(n>10)
	{
		cout<<"\n sorry.....You can Enter upto two vertices...";
		getch();
		exit(0);
	}
	for(i=1;i<=n;i++)
	for(j=1;j<=n;j++)
	a[i][j]=0;
	for(i=1;i<=n;i++)
	for(j=1;j<=n;j++)
	{
		if(i<j)
		{
			cout<<"\n Is There edge between
					Vertices"<<i<<"and"<<j<<"(1or0)";
			cin>>a[i][j];
			a[j][i]=a[i][j];
		}
	}
	for(i=1;i<=n;i++)
	visited[i]=0;
	cout<<"\n\n Output of Breadth First Serach\n\n";
	bfs(1);
	getch();
}
void bfs(int v)
{
	que q;
	visited[v]=1;
	if(v==1)
	cout<<v;
	else
	cout<<"--->"<<v; while(1)
	{
		for(int j=1;j<=n;j++)
		if(a[v][j]==1)
		if(visited[j]==0)
		{
			q.enqueue(j);
			visited[j]=1;
			cout<<"----- >"<<j;
		}
```

```
            v=q.dequeue();
            if(v==-1)
            break;
            }
}
```

OUTPUT:

***** BREADTH FIRST SEARCH *****

Enter the number of vertices in the graph: 3

Is there edge between vertices 1 and 2 (1 or 0) 0
Is there edge between vertices 1 and 3 (1 or 0) 1
Is there edge between vertices 2 and 3 (1 or 0) 1

Output of Breadth first search

1→ 3→ 2

EX.NO : 6

DEPTH FIRST SEARCH

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
int visited[10],n,a[10][10];
class que
{
        int a[10];
        int f,r;
        public:
        que()
        {
                f=r=0;
        }
        void enqueue(int);
        int dequeue();
};

void que::enqueue(int x)
{
        r=r+1;
        a[r]=x;
}
int que::dequeue()
{
        if(f!=r)
        {
                f=f+1;
                int x=a[f];
                return x;
        }
        else
        return-1;
}
void main()
{
        int i,j;
        void dfs(int v);
        clrscr();
        cout<<"\n\t\t***DEPTH FIRST SEARCH***\t\t\n";
        cout<<"\n Enter the number of vertices in the graph:";
```

```
        cin>>n;
        if(n>10)
        {
                cout<<"\n sorry.....You can Enter upto two vertices...";
                getch();
                exit(0);
        }
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        a[i][j]=0;
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
                if(i<j)
                {
                        cout<<"\n Is There edge between
                                        Vertices"<<i<<"and"<<j<<"(1or0)";
                        cin>>a[i][j];
                        a[j][i]=a[i][j];
                }
        }
        for(i=1;i<=n;i++)
        visited[i]=0;
        cout<<"\n\n output of Depth First Search\n\nm";
        dfs(1);
        getch();
}
void dfs(int v)
{
        int i,j;
        visited[v]=1;
        if(v==1) cout<<v;
        else
        cout<<"-->"<<v;
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        if(a[v][i]==1)
        if(visited[i]==0)
        dfs(i);
}
```

OUTPUT:

***** DEPTH FIRST SEARCH *****

Enter the number of vertices in the graph: 3

Is there edge between vertices 1 and 2 (1 or 0) 0
Is there edge between vertices 1 and 3 (1 or 0) 1
Is there edge between vertices 2 and 3 (1 or 0) 1

Output of Depth first search

1→ 3→ 2

EX.NO: 7

MERGE SORT

```
#include<iostream.h>
#include<conio.h>
void mergesort(int*,int,int);
void merge(int*,int,int);
void main()
{
        clrscr();
        int a[20],i,n;
        cout<<"\n\t MERGE SORT \t\n";
        cout<<"\n ------------------------";
        cout<<" Enter the number of elements \n";
        cin>>n:
        cout<<" Enter the element : \n";
        for(i=0;i<n;i++)
        cin>>a[i];
        mergesort(a,0,n-1);
        cout<<" Sorted element"<<endl;
        for(i=0;i<n;i++)
        cout<<a[i]<<endl;
        getch();
}
void mergesort(int a[],int first,int last)
{
        int mid;
        if(first<last)
        {
                mid=(first+last)/2;
                mergesort(a,first,mid);
                mergesort(a,mid+1,last)
                merge(a,first,last);
        }
}
void merge(int a[],int first,int last)
{
        int mid,l,m,n,k;
        int temp[20];
        mid=(first+last)/2;
        l=first;
        m=first;
        n=mid+1;
```

```
        while(m<=mid && n<=last)
        if(a[m]<a[n])
        temp[l++]=a[m++];
        else
        temp[l++]=a[n++];
        if(m>mid)
        for(k=n;k<=last;k++)
        {
                temp[l++]=a[k];
        }
        else
        for(k=m;k<=mid;k++)
        {
                temp[l++]=a[k];
        }
        cout<<"\n";
        for(k=first;k<=last;k++)
        {
                a[k]=temp[k];
                cout<<a[k]<<" ";
        }
}
```

OUTPUT:

MERGE SORT

Enter the number of elements :

4

Enter the element :

4
6
5
7
sorted element is : 4 5 6 7

EX.NO : 8

QUICK SORT

```
#include<iostream.h>
#include<conio.h>
void quicksort(int*,int,int)
void main()
{
        clrscr();
        int a[20],i,n;
        cout<<"\n\t QUICK SORT \t\n";
        cout<<"\n ------------------------";
        cout<<" Enter the number of elements \n";
        cin>>n:
        cout<<" Enter the element : \n";
        for(i=0;i<n;i++)
        cin>>a[i];
        quicksort(a,0,n-1);
        cout<<" Sorted element"<<endl;
        for(i=0;i<n;i++)
        cout<<a[i]<<endl;
        getch();
}
void quicksort(int a[],int lo,int hi)
{
        int p,temp;
        if(lo<hi)
        {
                p=a[lo];
                int i=lo;
                int j=hi;
                while(i<j)
                {
                        while(a[i]<=p && i<j)
                        i++;
                        while(a[j]>p && I<=j)
                        j--;
                        if(i<=j)
                        {
                                temp=a[i];
                                a[i]=a[j];
                                a[j]=temp;
                                cout<<endl;
```

```
                        }
                }
        temp=a[j];
        a[j]=a[lo];
        a[lo]=temp;
        for(i=0;i<hi;i++)
        cout<<a[i]<<endl;
        quicksort(a,lo,j-1);
        quicksort(a,j+1,hi)
        }
}
```

OUTPUT:

QUICK SORT

Enter the number of elements :

5

Enter the element :

86
34
93
27
63

sorted element : 27 34 63 86 93