Assignment No 4

Name :- Amit Bandu Swami
Roll No:- 2221018

Problem :- Implement graph representation using adjacency matrix and adjacency list & BFS,DFS traversals

```
#include <iostream>
using namespace std;
class node
{
        public:
        int data ;
        node *next;
};
class graph
{
        public:
        int nv,ne,a[10][10],b[10][10];
        node *head[10];
        void adj_matrix();
        void adj_list();
        void bfs(int v,int visit[]);
        void dfs(int v,int visit[]);

};

class queue
{
        public:
                int f;int r;int s[10];
                queue()
                {
                        f=-1;r=-1;
                }
                void enqueue(int v )
                {
                        r=r+1;
                        s[r]=v;
                }
                int dequeue()
                {
                        f=f+1;
                        int k=s[f];
                        return k;
                }
```

```
};

void graph::adj_matrix()
{
	int m,n;
	cout<<"Enter no of vertices :";
	cin>>nv;
	cout<<"Enter no of edges :";
	cin>>ne;
	for(int i=0;i<nv;i++)
	{
		for(int j=0;j<nv;j++)
		{
			a[i][j]=0;
		}
	}
	for(int i=0;i<ne;i++)
	{
		cout<<"Enter start and end vertices :";
		cin>>b[i][0]>>b[i][1];
	}
	for(int i=0;i<ne;i++)
	{
		m=b[i][0];
		n=b[i][1];
		a[m][n]=1;
	}
	for(int i=0;i<nv;i++)
	{
		for(int j=0;j<nv;j++)
		{
			cout<<a[i][j]<<"  ";
		}
		cout<<endl;
	}
}

void graph::adj_list()
{
	node *n,*p,*temp,*prev;
	int k,l;
	for(int i=0;i<nv;i++)
	{
		head[i]=new node;
		head[i]->data=i;
		head[i]->next=NULL;
	}
	for(int i=0;i<ne;i++)
```

```
		{
			k=b[i][0];
			l=b[i][1];
			n=new node;
			n->data=l;
			n->next=NULL;
			prev=head[k];
			temp=prev->next;
			while(temp!=NULL && temp->data<l)
			{
				prev=prev->next;
				temp=prev->next;
			}
			n->next=prev->next;
			prev->next=n;
		}
		for(int i=0;i<nv;i++)
		{
		p=head[i];
		while(p!=NULL)
		{
			cout<<p->data<<" ->";
			p=p->next;

		}
		cout<<endl;

		}
}
void graph::dfs(int v,int visit[])
{
		visit[v]=1;
		cout<<head[v]->data<<" ->";
		node *temp;
		temp=head[v]->next;
		while(temp!=NULL)
		{
			if(visit[temp->data]!=1)
			{
				dfs(temp->data,visit);
			}
			temp=temp->next;
		}
		cout<<endl;
}
void graph::bfs(int v,int visit[20])
{
		queue q;
```

```
        int t;
        node *temp;
        q.enqueue(v);
        visit[v]=1;
        cout<<head[v]->data<<" ";
        q.enqueue(-1);
        cout<<endl;
        while(q.f!=q.r)
                {
                        t=q.dequeue();
                        if(t!=-1)
                        {
                                temp=head[t]->next;
                                while(temp!=NULL)
                                {
                                        if(visit[temp->data]!=1)
                                        {
                                                q.enqueue(temp->data);
                                                visit[temp->data]=1;
                                                cout<<temp->data<<" ";
                                        }
                                        temp=temp->next;
                                 }
                         }

                        else
                        {
                                if(q.f!=q.r)
                                {
                                         q.enqueue(-1);
                                         cout<<endl;
                                }
                        }
                }
}


int main()
{
        graph ob;
        int ch,v,visit[10];
        while(1)
        {
                cout<<"1. Adjacency Matrix \n";
                cout<<"2. Adjacency list \n";
                cout<<"3. DFS \n";
                cout<<"4. BFS \n";
                cout<<" Enter Your choice :";
```

```
                cin>>ch;
                switch(ch)
                {
                        case 1:ob.adj_matrix();
                                break;
                        case 2:ob.adj_list();
                                break;
                        case 3:cout<<"Enter starting vertex :";
                                cin>>v;
                                for(int i=0;i<ob.nv;i++)
                                {
                                        visit[i]=0;
                                }
                                ob.dfs(v,visit);
                                break;
                        case 4:cout<<"Enter starting vertex :";
                                cin>>v;
                                for(int i=0;i<ob.nv;i++)
                                {
                                        visit[i]=0;
                                }
                                ob.bfs(v,visit);
                }

        }


}
```