```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int vertex;
    struct node *link;
}*gptr[20],*ptr,*temp,*head;
struct stack
{
    int item;
    struct stack *nxt;
}*top=NULL;
void push(int);
int pop();
void push(int pt)
{
    struct stack *tp=(struct stack *)malloc(sizeof(struct
stack));
    tp->item=pt;
    tp->nxt=NULL;
    if(top==NULL)
        top=tp;

    else
    {
        tp->nxt=top;
        top=tp;
    }
}
int pop()
{
    if(top!=NULL)
    {
        int del=top->item;
        top=top->nxt;
        return del;
    }
}
struct queue
{
    int item;
    struct queue *nxt;
}*front=NULL,*rear=NULL;
void enqueue(int);
int dequeue();
```

```
void enqueue(int pt)
{
    struct queue *tp=(struct queue *)malloc(sizeof(struct
queue));
    tp->item=pt;
    tp->nxt=NULL;
    if(front==NULL)
    {
       front=tp;
       rear=tp;
    }

    else
    {
        rear->nxt=tp;
        rear=tp;
    }
}
int dequeue()
{
    if(front!=NULL)
    {
        int del=front->item;
        front=front->nxt;
        return del;
    }
}
int vdfs[50],vbfs[50],n,ind1=0,ind2=0;
int search1(int b);
int search1(int b)
{
    int flag=0;
    for(int i=0;i<n;i++)
    {
        if(vdfs[i]==b)
        {
            flag=1;
            return 1;
        }
    }
    if(flag==0)
    {
        return 0;
    }
}
```

```
void insert1(int b);
void insert1(int b)
{
    vdfs[ind1++]=b;
}
int search2(int b);
int search2(int b)
{
    int flag=0;
    for(int i=0;i<n;i++)
    {
        if(vbfs[i]==b)
        {
            flag=1;
            return 1;
        }
    }
    if(flag==0)
    {
        return 0;
    }
}
void insert2(int b);
void insert2(int b)
{
    vbfs[ind2++]=b;
}
void dfs();
void bfs();
void main()
{
    int av,ch,vert;
    printf("How many vertices are there? ");
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        gptr[i]=NULL;
        head=NULL;
        printf("How many adjacent vertices of vertex %d: ",i);
        scanf("%d",&av);
        printf("Enter the adjacent vertices\n");
        for(int j=0;j<av;j++)
        {
            temp=(struct node *)malloc(sizeof(struct node));
            scanf("%d",&vert);
```

```
        temp->vertex=vert;
        temp->link=NULL;
        if(gptr[i]==NULL)
        {
            gptr[i]=temp;
            head=temp;
            //printf("%d\n",head->vertex);
        }
        else
        {
            head->link=temp;
            head=temp;
            //printf("%d\n",head->vertex);
        }
    }
}
while(1)
{
    printf("**Traversal**\n");
    printf("1.(DFS)Depth first search\n");
    printf("2.(BFS)Breadth first search\n");
    printf("3.Display vertices\n");
    printf("4.Exit\n");
    printf("Enter your choice: ");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
            dfs();
            printf("\n");
            break;

        case 2:
         bfs();
         printf("\n");
         break;

        case 3:
            for(int i=1;i<=n;i++)
            {
                ptr=gptr[i];
                while(ptr!=NULL)
                {
                    printf("Vertex %d\n",ptr->vertex);
                    ptr=ptr->link;
```

```
                    }
                }
                printf("\n");
                break;

            case 4:
                exit(0);

            default:
                printf("Invalid choice\n");
        }
    }
}
void dfs()
{
    if(gptr[1]==NULL)
    {
        printf("The graph is empty\n");
    }
    else
    {
        int u=1;
        push(u);
        while(top!=NULL)
        {
            u=pop();
            if(search1(u)==0)
            {
                insert1(u);
                //printf("Vertex is %d\n",u->vertex);
                ptr=gptr[u];
                while(ptr!=NULL)
                {
                    push(ptr->vertex);
                    //printf("Adjacent vertex is %d\n",ptr-
>vertex);
                    ptr=ptr->link;
                }
            }
        }
        printf("Displaying (DFS)Depth first search: ");
        for(int i=0;i<n;i++)
        {
            printf("%d\t",vdfs[i]);
        }
```

```
        printf("\n");
    }
}
void bfs()
{
    if(gptr[1]==NULL)
    {
        printf("The graph is empty\n");
    }
    else
    {
        int u=1;
        enqueue(u);
        while(front!=NULL)
        {
            u=dequeue();
            if(search2(u)==0)
            {
                insert2(u);
                //printf("Vertex is %d\n",u->vertex);
                ptr=gptr[u];
                while(ptr!=NULL)
                {
                    enqueue(ptr->vertex);
                    //printf("Adjacent vertex is %d\n",ptr-
>vertex);
                    ptr=ptr->link;
                }
            }
        }
        printf("Displaying (BFS)Breadth first search : ");
        for(int i=0;i<n;i++)
        {
            printf("%d\t",vbfs[i]);
        }
        printf("\n");
    }
}
/*
OUTPUT:
How many vertices are there? 8
How many adjacent vertices of vertex 1: 3
Enter the adjacent vertices
2 3 8
How many adjacent vertices of vertex 2: 3
```

```
Enter the adjacent vertices
1 4 5
How many adjacent vertices of vertex 3: 3
Enter the adjacent vertices
1 6 7
How many adjacent vertices of vertex 4: 2
Enter the adjacent vertices
2 8
How many adjacent vertices of vertex 5: 2
Enter the adjacent vertices
8 3
How many adjacent vertices of vertex 6: 2
Enter the adjacent vertices
8 3
How many adjacent vertices of vertex 7: 2
Enter the adjacent vertices
8 3
How many adjacent vertices of vertex 8: 5
Enter the adjacent vertices
4 5 6 1 7
**Traversal**
1.(DFS)Depth first search
2.(BFS)Breadth first search
3.Display vertices
4.Exit
Enter your choice: 1
Displaying (DFS)Depth first search: 1  8
   7  3  6  5  4  2

**Traversal**
1.(DFS)Depth first search
2.(BFS)Breadth first search
3.Display vertices
4.Exit
Enter your choice: 2
Displaying (BFS)Breadth first search : 1
   2  3  8  4  5  6  7

**Traversal**
1.(DFS)Depth first search
2.(BFS)Breadth first search
3.Display vertices
4.Exit
Enter your choice: 4
*/
```