NAME: Vishal J Lodha

SRN: PES1UG20CS507

Section: P

header.h

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<stdlib.h>
#define MAX 100
typedef struct node
{
   int vid;
   int loci;
   int locj;
   struct node *link;
}NODE;
typedef struct nodedemo
{

   int loci[100];
   int locj[100];
   int vid[100];
}node_demo;

typedef struct queue
{
   int from[MAX];
   int to[MAX];
   int front;
   int rear;
```

```
}QUEUE;
typedef struct Graph
{
  int numVertices;
  NODE *h[100];
  int visited[100];
}GRAPH;
int er,ec,sr,sc;
int pathdfs[MAX];
int len;
int read(int a[10][10]);
int create_matrix(int a[10][10],node_demo* info,int adj_mat[MAX][MAX],int n);
void initgraph(GRAPH *g,int n);
void convert_list(int a[MAX][MAX],GRAPH* g,node_demo* info);
int pathtraversalbfs(GRAPH *g, int startVertex,int destination,int pathbfs[MAX]);
int pathtraversaldfs(GRAPH *g, int startvertex,int destinationvertex);
void displaybfs(node_demo* info,int pathbfs[MAX],int n);
void displaydfs(node_demo* info);
```

server.c

```
NODE *getnode(int vid,int loci,int locj)
{
    NODE * temp;
    temp=(NODE *)malloc(sizeof(NODE));
    temp->vid=vid;
    temp->loci=loci;
    temp->locj=locj;
    temp->link=NULL;
    return(temp);
}
void initgraph(GRAPH *g,int n)
{
```

```
    g->numVertices=n;
    for(int i=0;i<n;i++)
    {
        g->visited[i]=0;
        g->h[i]=NULL;
    }
}
int create_matrix(int a[10][10],node_demo* info,int adj_mat[MAX][MAX],int n)
{

    int i,j;
    int ini=0;
    int c=0;

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(a[i][j]==0)
            {
                info->loci[c]=i;
                info->locj[c]=j;
                info->vid[c]=c+1;
                c=c+1;
            }
        }
    }
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
```

```
if(a[i][j]==0)
{
    if (a[i][j+1]==0)
    {
        int check=-1;
        for (int k = 0; k<c; k++)
        {
            if (info->loci[k]==i && info->locj[k]==j+1)
            {
                check=k;
                break;
            }

        }
        if (check!=-1)
        {
            adj_mat[ini][check]=1;
            adj_mat[check][ini]=1;
        }
    }
    if (a[i+1][j]==0)
    {
        int check=-1;
        for (int k = 0; k<c; k++)
        {
            if (info->loci[k]==i+1 && info->locj[k]==j)
            {
                check=k;
                break;
            }
```

```
                }
                if (check!=-1)
                {
                    adj_mat[ini][check]=1;
                    adj_mat[check][ini]=1;
                }
            }
        ini =ini+1;
        }


      }
    }
    return c;
}
int read(int a[10][10])
{
            FILE *ptr;
            ptr = fopen("input.txt","r");
            if(ptr == NULL)
            {
     printf("Error!");
     exit(1);
            }
            fscanf(ptr,"%d%d%d%d",&sr,&sc,&er,&ec);
            for (int i = 0; i < er+1; i++)
    {
      for (int j = 0; j < ec+1; j++)
      {
        char c;
        if (fscanf(ptr, " %c", &c) != 1)
            printf("...report read failure and exit...");
```

```
            else if (isdigit((unsigned char)c))
                a[i][j] = c - '0';
            else
                a[i][j] = 0;
        }
    }
    return ec;
}
void convert_list(int a[MAX][MAX],GRAPH* g,node_demo* info)
{
    NODE *ptr,*temp;
    int i,j;
    for(i=0;i<g->numVertices;i++)
    {
        for(j=0;j<g->numVertices;j++)
        {
            if(a[i][j]==1)
            {
                temp=getnode(j,info->loci[j],info->locj[j]);
                if(g->h[i]==NULL)
                    g->h[i]=temp;
                else
                {
                    ptr=g->h[i];
                    while(ptr->link!=NULL)
                    {
                        ptr=ptr->link;
                    }
                    ptr->link=temp;
                }
            }
```

```
        }
    }
}
void initQueue(QUEUE* q)
{
    q->rear = -1;
    q->front = -1;
}
void bfs(GRAPH *g, int startVertex,QUEUE* q)
{
    int s=startVertex;
    g->visited[startVertex] = 1;
    q->front=0;
    q->rear=0;
    q->to[0]=startVertex;
    q->from[0]=-1;
    do
    {
        s=q->to[q->front];
        NODE* temp=g->h[s];
        while (temp!=NULL )
        {
            if (g->visited[temp->vid]==0)
            {
                q->rear++;
                q->from[q->rear]=s;
                q->to[q->rear]=temp->vid;
                g->visited[temp->vid] = 1;
            }
            temp=temp->link;
        }
```

```
        q->front++;


    } while (q->front<=q->rear);
}
int pathtraversalbfs(GRAPH *g, int startVertex,int destination,int pathbfs[MAX])
{
    int k=-1,c=0;
    QUEUE q;
    initQueue(&q);
    bfs(g,startVertex,&q);
    for (int i = 0; i < q.rear+1; i++)
    {
        if (q.to[i] == destination)
        {
            k=i;
            break;
        }
    }
    if (k!=-1)
    {
        pathbfs[c++]=q.to[k];
        while (k!=0)
        {


            for (int i = 0; i < k; i++)
            {
                if (q.to[i] == q.from[k])
                {
                    k=i;
                    break;
                }
```

```
            }
            pathbfs[c++]=q.to[k];
        }
        pathbfs[c]=q.to[0];
        return c;
    }
    else
    {
        return 0;
    }
}
void displaybfs(node_demo* info,int pathbfs[MAX],int n)
{

            FILE* ptr = fopen("outbfs.txt","w");
            for (int i = n-1; i >=0; i--)
    {
                        fprintf(ptr,"%d %d\n",info->loci[pathbfs[i]],info->locj[pathbfs[i]]);
            }
            fclose(ptr);
}
void displaydfs(node_demo* info)
{

            FILE* ptr = fopen("outdfs.txt","w");
            for (int i = 0; i <= len; i++)
    {
                        fprintf(ptr,"%d %d\n",info->loci[pathdfs[i]],info->locj[pathdfs[i]]);
            }
            fclose(ptr);
}
```

```
void initvisited(int visited[MAX],int n)
{
   for(int i=0;i<n;i++)
   {
      visited[i]=0;
   }
}
int DFS(GRAPH* g, int startvertex,int destination)
{
   NODE* temp = g->h[startvertex];
   g->visited[startvertex] = 1;
   while (temp != NULL)
   {

      if (g->visited[temp->vid] == 0)
      {
         len++;
         pathdfs[len]=temp->vid;
        if((temp->vid==destination)|| DFS(g,temp->vid,destination))
        {
           return 1;
        }
      }
      temp = temp->link;
   }
   --len;
   return 0;
}
int pathtraversaldfs(GRAPH *g, int startvertex,int destinationvertex)
{
```

```
    initvisited(g->visited,g->numVertices);
    pathdfs[len] = startvertex;
    int c=DFS(g,startvertex,destinationvertex);
    return c;
}
```

Client.c

```
int main()
{
    node_demo info;
    GRAPH g;
    int adj_mat[MAX][MAX]={0};
    int a[10][10];
    int pathbfs[MAX]={0};
    int pathdfs[MAX]={0};
    int n=read(a)+1;
    int c = create_matrix(a,&info,adj_mat,n);
    initgraph(&g,c);
    convert_list(adj_mat,&g,&info);
    n=pathtraversalbfs(&g,0,c-1,pathbfs);
    if(n)
    {
        displaybfs(&info,pathbfs,n);
    }
    else
    {
        FILE* ptr = fopen("outbfs.txt","w");
                        fprintf(ptr,"-1");
                fclose(ptr);
    }
    c=pathtraversaldfs(&g,0,c-1);
```

```
    if(n)
    {
        displaydfs(&info);
    }
    else
    {
        FILE* ptr = fopen("outdfs.txt","w");
                        fprintf(ptr,"-1");
                fclose(ptr);
    }
}
```

Input.txt

```
0 0
9 9
0 0 0 0 0 1 1 0 0 0
1 1 0 1 0 0 1 0 1 0
1 1 0 1 0 0 0 1 1 0
0 1 0 0 0 1 0 0 1 0
1 1 1 0 1 1 1 0 1 0
0 1 0 0 0 1 1 0 0 1
1 1 1 1 0 1 1 0 1 0
1 0 0 0 0 0 0 0 1 1
0 0 0 0 0 1 1 0 0 0
1 1 0 1 1 0 0 1 1 0
```

Outbfs.txt

outbfs.txt

```
0 0
0 1
0 2
0 3
0 4
1 4
1 5
2 5
2 6
3 6
3 7
4 7
5 7
6 7
7 7
8 7
8 8
8 9
9 9
```

Outdfs.txt

outu3.txt

```
0 0
0 1
0 2
0 3
0 4
1 4
1 5
2 5
2 4
3 4
3 3
4 3
5 3
5 4
6 4
7 4
7 5
7 6
7 7
8 7
8 8
8 9
9 9
```