/*

NAME-YASH GANDHI SE IT BATCH A 14

QUESTION-
An airline company is interested in airline route a mong seven cities: Delhi, Mumbai, Jaipur, Pune, Bangalore, Ahmedabad and Goa. It flies on the following routes:

a) Ahmedabad to Goa h) Jaipur to Ahmedabad

b) Ahmedabad to Pune i) Jaipur to Delhi

c) Bangalore to Jaipur j) Jaipur to Goa

d) Bangalore to Pune k) Mumbai to Pune

e) Delhi to Jaipur l) Pune to Bangalore

f) Delhi to Mumbai

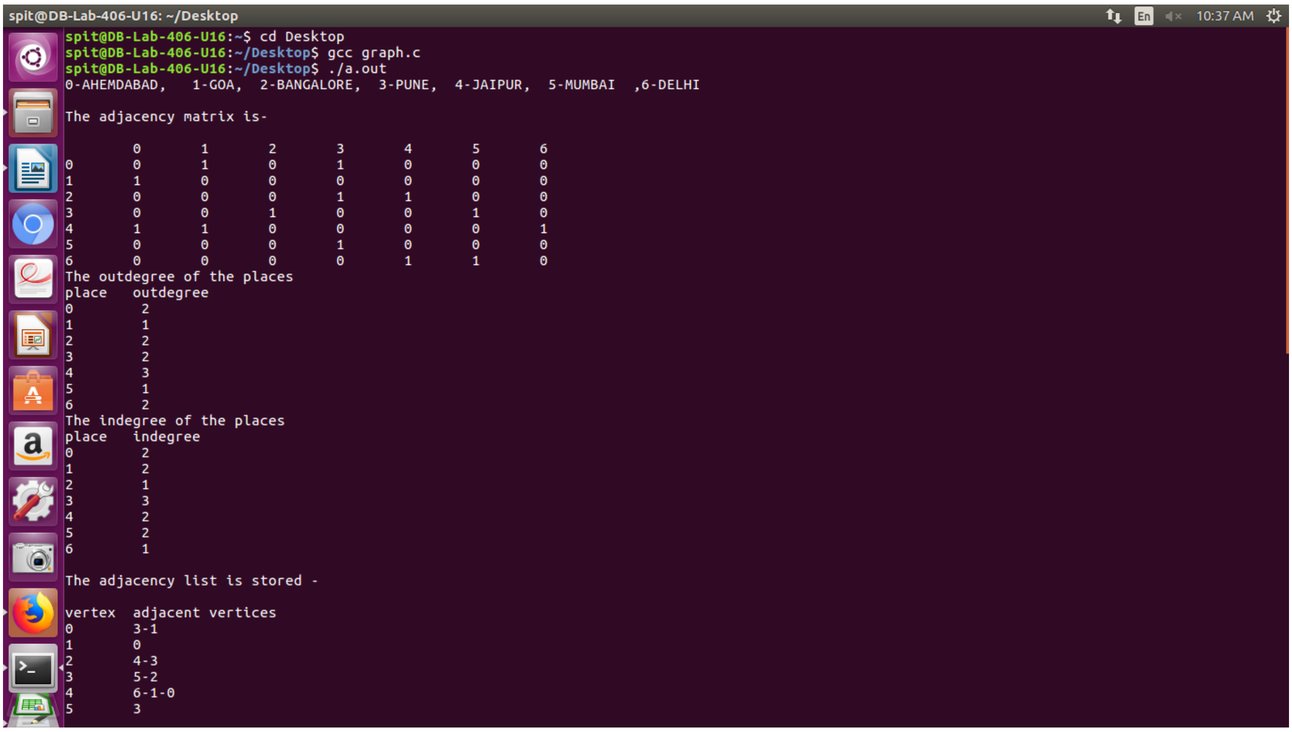
g) Goa to Ahmedabad m) Pune to Mumbai

Questions:

a. Represent the above scenario using Graph.Represent the Graph in Adjacency matrix and Adjacency List represenation both

-For creating Adjacency list take adjacency matrix as input

The Graph function is used for it- The adjacency list is shown below



```
spit@DB-Lab-406-U16:~$ cd Desktop
spit@DB-Lab-406-U16:~/Desktop$ gcc graph.c
spit@DB-Lab-406-U16:~/Desktop$ ./a.out
0-AHEMDABAD,   1-GOA,  2-BANGALORE,  3-PUNE,  4-JAIPUR,  5-MUMBAI  ,6-DELHI

The adjacency matrix is-

        0       1       2       3       4       5       6
0       0       1       0       1       0       0       0
1       1       0       0       0       0       0       0
2       0       0       0       1       1       0       0
3       0       0       1       0       0       1       0
4       1       1       0       0       0       0       1
5       0       0       0       1       0       0       0
6       0       0       0       0       1       1       0
The outdegree of the places
place   outdegree
0        2
1        1
2        2
3        2
4        3
5        1
6        2
The indegree of the places
place   indegree
0        2
1        2
2        1
3        3
4        2
5        2
6        1

The adjacency list is stored -

vertex  adjacent vertices
0       3-1
1       0
2       4-3
3       5-2
4       6-1-0
5       3
```

- Show indegree and out degree of each vertex

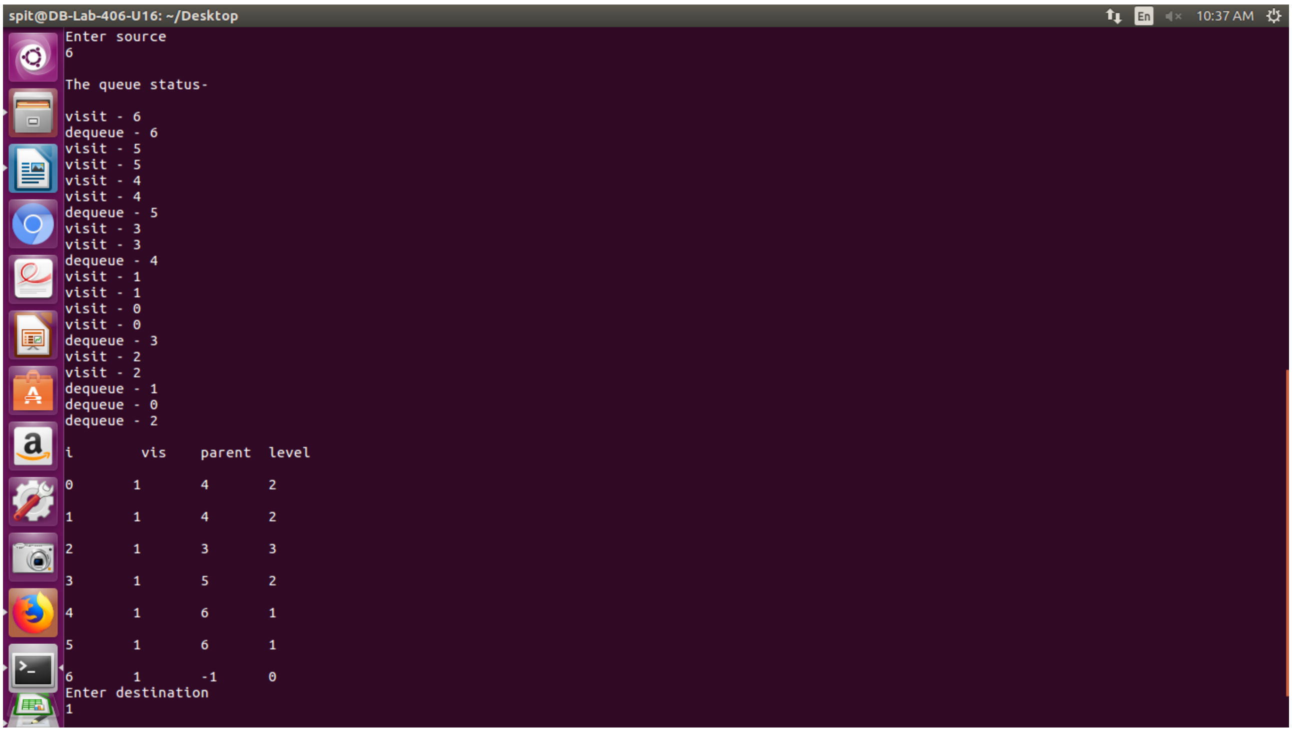
The get_degree function is used for it

The indegree and outdegree are displayed in the above picture

b. Design, apply and implement a strategy to find the route (direct or indirect) from one city to another city.

The bfs function is used for it-

the queue status is displayed

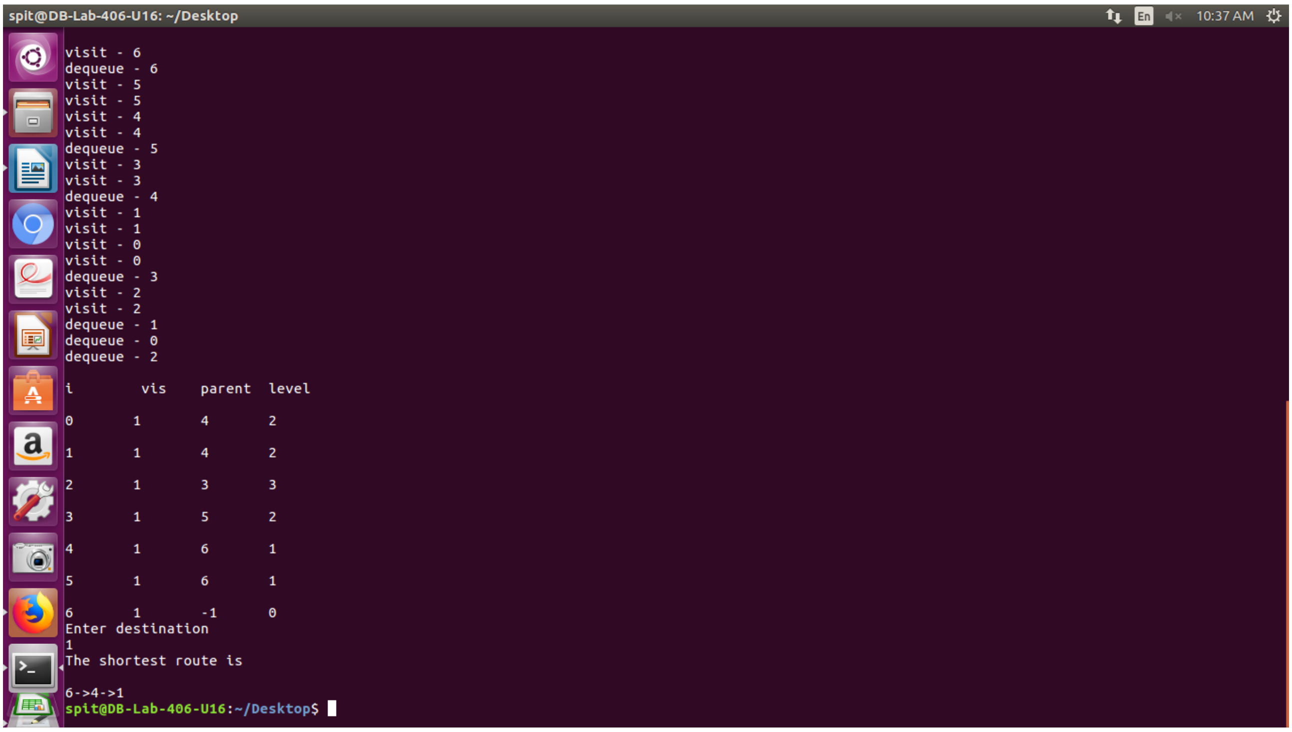


```
Enter source
6

The queue status-

visit - 6
dequeue - 6
visit - 5
visit - 5
visit - 4
visit - 4
dequeue - 5
visit - 3
visit - 3
dequeue - 4
visit - 1
visit - 1
visit - 0
visit - 0
dequeue - 3
visit - 2
visit - 2
dequeue - 1
dequeue - 0
dequeue - 2

i        vis     parent  level

0        1       4       2

1        1       4       2

2        1       3       3

3        1       5       2

4        1       6       1

5        1       6       1

6        1       -1      0
Enter destination
1
```

c. Is there any route from Delhi to Goa?

Yes the route is Delhi-Jaipur-Goa

route is displayed int the picture in the form of keys mention 6->4->1



```
visit - 6
dequeue - 6
visit - 5
visit - 5
visit - 4
visit - 4
dequeue - 5
visit - 3
visit - 3
dequeue - 4
visit - 1
visit - 1
visit - 0
visit - 0
dequeue - 3
visit - 2
visit - 2
dequeue - 1
dequeue - 0
dequeue - 2

i         vis     parent  level

0       1       4       2

1       1       4       2

2       1       3       3

3       1       5       2

4       1       6       1

5       1       6       1

6       1       -1      0
Enter destination
1
The shortest route is

6->4->1
spit@DB-Lab-406-U16:~/Desktop$ 
```

```
*/


#include<stdio.h>
#include<stdlib.h>

typedef struct node              //structure declaration for node
{
 int vertex;
 struct node* next;
}node;

node* head[10];             //global declare-arrays and head array
int visited[10];
int level[10];
int parent[10];

void printlist()              //function to print adj list
{
  node* curr;
  int i,j;
  printf("vertex\tadjacent vertices\n");
  for(i=0;i<7;i++)
  {
    printf("%d\t",i);
    curr=head[i];
    while(curr->next!=NULL)
     {
       printf("%d-",curr->vertex);
       curr=curr->next;
     }
     printf("%d\n",curr->vertex);
    }

}



void graph(int a[][7])            //function to form adj list
{
int i,j;
 for(i=0;i<7;i++)
  {
  head[i]=NULL;
  for(j=0;j<7;j++)
   {
     if(a[i][j]==1)
     {
      node* newnode=(node*)malloc(sizeof(node*));
      newnode->vertex=j;
      if(head[i]==NULL)
      {
           head[i]=newnode;
           newnode->next=NULL;
      }
      else
      {
       newnode->next=head[i];
       head[i]=newnode;

      }
     }

   }
  }
printf("\n");
printlist();
printf("\n");
}


typedef struct qnode            //queue nodes structure
{
    int vertex;
    struct qnode* next;
}qnode;
                                                  /*Queue ADT*/
typedef struct queue
{
    qnode* front, *rear;
}queue;

void  enq(queue*q,int v)                               //enqueue
```

```
{
    qnode* newnode=(qnode*)malloc(sizeof(qnode*));
    newnode->vertex=v;
    newnode->next=NULL;
    if(q->front==NULL)
    {
       q->front=newnode;
        q->rear=newnode;
    }
    else
    {
        q->rear->next=newnode;
        q->rear=newnode;
    }
}

int deq(queue*q)                        //dequeue
{
     int p;
    qnode* temp;
    if(q->front==NULL)
        printf("\nQueue Underflow! Can't Delete!");
   else
    {
        temp=q->front;
        q->front=q->front->next;
        p=temp->vertex;
        free(temp);
        if(q->front==NULL)
            q->rear=NULL;
    }
return p;
}
int empty(queue* q)           //is empty
{
  if(q->front==NULL)
   return 1;
 else
  return  0;
}

void print()                  //print
{
  int i;
  printf("\ni\t vis\tparent\tlevel\n");
  for(i=0;i<7;i++)
 {
   printf("\n%d \t%d \t%d \t%d\n",i,visited[i],parent[i],level[i]);
 }
}
```

```
void bfs(int x)                //breadth first search
{
  int s=x;
  queue q;
  q.front=NULL;
  q.rear=NULL;
  int i,m;
  for(i=0;i<7;i++)            //common value to all vertice
   {
    visited[i]=0;
    parent[i]=-1;
    level[i]=-1;

   }
  enq(&q,s);
  visited[s]=1;
  printf("\nvisit - %d\n",s);
  parent[s]=-1;;
  level[s]=0;
  node* curr;

  while(!empty(&q))
  {
       s=deq(&q);
        printf("dequeue - %d\n",s);
      for(curr=head[s];curr!=NULL;curr=curr->next)
       {
         m=curr->vertex;
         if(visited[m]==0)
           {
               enq(&q,m);
                printf("visit - %d\n",m);
               parent[m]=s;
               visited[m]=1;
               level[m]=level[s]+1;
               printf("visit - %d\n",m);
           }
        }
   }
print();
}

void getroute(int destination)          //function to get a route
{
   int i=destination;
   int a[7];
   int j=1,m;
    a[0]=i;
  while(parent[i]!=-1)
  {
    a[j]=parent[i];
    i=parent[i];
    j=j+1;
   }
printf("\n");
  for(m=j-1;m>0;m--)
  {
   printf("%d->",a[m]);
  }
  printf("%d\n",a[m]);
}

void get_degree(int a[][7])    //function to get degree
{
 int b[7];
 int sum=0;
 int i,j;

  printf("The outdegree of the places\n");
 for(i=0;i<7;i++)
 {
  for(j=0;j<7;j++)
  {
   if(a[i][j]==1)
   sum=sum+1;
  }
  b[i]=sum;
  sum=0;
 }
  printf("place\toutdegree\n");

  for(i=0;i<7;i++)
  {
    printf("%d\t %d\n",i,b[i]);
  }

    printf("The indegree of the places\n");
   sum=0;
   for(j=0;j<7;j++)
  {
  for(i=0;i<7;i++)
   {
   if(a[i][j]==1)
   sum=sum+1;
   }
  b[j]=sum;
  sum=0;
  }

  printf("place\tindegree\n");

  for(j=0;j<7;j++)
  {
    printf("%d\t %d\n",j,b[j]);
  }



}


void main()
{
int i,j;
printf("0-AHEMDABAD,  1-GOA,  2-BANGALORE,  3-PUNE,  4-JAIPUR,  5-MUMBAI  ,6-
DELHI\n\n");//key
int a[7][7]={
             {0,1,0,1,0,0,0},
             {1,0,0,0,0,0,0},
             {0,0,0,1,1,0,0},
             {0,0,1,0,0,1,0},
             {1,1,0,0,0,0,1},
             {0,0,0,1,0,0,0},
             {0,0,0,0,1,1,0},
            };

            printf("The adjacency matrix is-\n\n");  //print matrix
            printf("\t0\t1\t2\t3\t4\t5\t6\n");
    for(i=0;i<7;i++)
    {
     printf("%d\t",i);
     for(j=0;j<7;j++)
     {
     printf("%d\t",a[i][j]);
     }
     printf("\n");
    }

  get_degree(a);

  printf("\nThe adjacency list is stored -\n");
  graph(a);                            //forming adj list
 int x;
 printf("Enter source\n");
 scanf("%d",&x);
 printf("\nThe queue status-\n");
 bfs(x);                               //performing bfs
 printf("Enter destination \n");
 scanf("%d",&x);
 printf("The shortest route is\n");
 getroute(x);                          //getting route to destination

 }
/*
OUTPUT-
/*
```

```
spit@DB-Lab-406-U16:~$ cd Desktop
spit@DB-Lab-406-U16:~/Desktop$ gcc graph.c
spit@DB-Lab-406-U16:~/Desktop$ ./a.out
0-AHEMDABAD,   1-GOA,  2-BANGALORE,  3-PUNE,  4-JAIPUR,  5-MUMBAI  ,6-DELHI

The adjacency matrix is-

        0       1       2       3       4       5       6
0       0       1       0       1       0       0       0
1       1       0       0       0       0       0       0
2       0       0       0       1       1       0       0
3       0       0       1       0       0       1       0
4       1       1       0       0       0       0       1
5       0       0       0       1       0       0       0
6       0       0       0       0       1       1       0
The outdegree of the places
place   outdegree
0        2
1        1
2        2
3        2
4        3
5        1
6        2
The indegree of the places
place   indegree
0        2
1        2
2        1
3        3
4        2
5        2
6        1

The adjacency list is stored -

vertex  adjacent vertices
0       3-1
1       0
2       4-3
3       5-2
4       6-1-0
5       3
6       5-4

Enter source
6

The queue status-

visit - 6
dequeue - 6
```

```
visit - 5
visit - 5
visit - 4
visit - 4
dequeue - 5
visit - 3
visit - 3
dequeue - 4
visit - 1
visit - 1
visit - 0
visit - 0
dequeue - 3
visit - 2
visit - 2
dequeue - 1
dequeue - 0
dequeue - 2

i        vis     parent  level

0        1       4       2

1        1       4       2

2        1       3       3

3        1       5       2

4        1       6       1

5        1       6       1

6        1       -1      0
Enter destination
1
The shortest route is

6->4->1

*/
```