NAME-YASH GANDHI BATCH-A SE IT ROLL NO 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

b) Ahmedabad to Pune

c) Bangalore to Jaipur

d) Bangalore to Pune

e) Delhi to Jaipur

f) Delhi to Mumbai

g) Goa to Ahmedabad

h) Jaipur to Ahmedabad

i) Jaipur to Delhi

j) Jaipur to Goa

k) Mumbai to Pune

l) Pune to Bangalore

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

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

c. Is there any route from Delhi to Goa?

DFS IMPLEMENTATION CODE

```
#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");

}

void print()

{

  int i;

  printf("\n\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 getroute(int destination)

{

   int i=destination;

   int a[7];

   int j=1,m;

    a[0]=i;

   printf("\nthe route is\n");

  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]);

}


typedef struct snode     //Structure for node of linked list

{

             int data;

             struct snode *next;

}snode;
```

```
typedef struct STACK    //Structure for stack
{
        snode *last;
}stack;

void push(stack *p,int data)    //Function to push data on to the stack
{
        snode *newnode;
        newnode=(snode*)malloc(sizeof(snode));
        newnode->next=NULL;
        if(newnode==NULL)
                printf("Stack is full\n");
        if(p->last==NULL)
        {
                newnode->data=data;

                p->last=newnode;
        }
        else
        {
          newnode->data=data;
          newnode->next=p->last;
          p->last=newnode;
        }
}
```

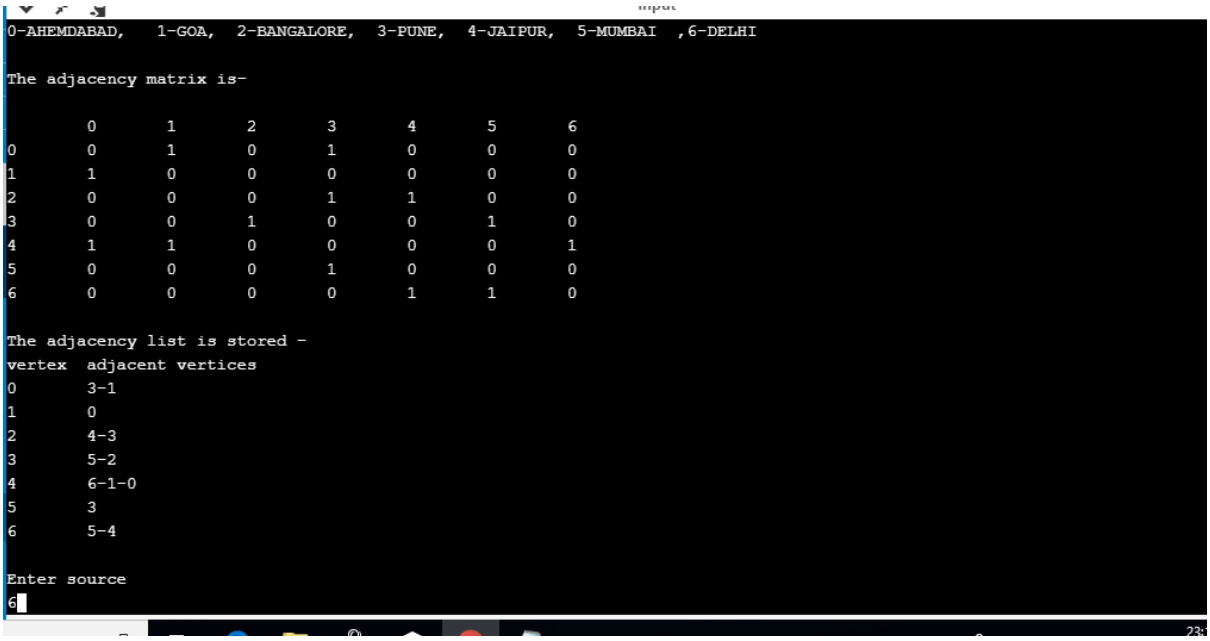
```
int pop(stack *p)               //Function to pop data from the stack

{

        if(p->last==NULL)       //Checks if stack is empty

        {

                printf("Stack is empty\n");

                return -1;

        }

        if(p->last->next==NULL)

         {

            snode *temp;

                temp=p->last;

                int v=temp->data;

                p->last=NULL;

                free(temp);

                return v;

         }

        else

        {

                snode *temp;

                temp=p->last;

                int v=temp->data;

                p->last=temp->next;

                free(temp);

                return v;

        }
```

```
}


int isstackempty(stack* p)
{
 if(p->last==NULL)
 return 1;
 else
 return 0;
}
void dfs(int s,stack p)     //DFS FUNCTION
{
   node* ptr=head[s];
   if(visited[s]==0)
 {
  visited[s]=1;
  push(&p,s);
  printf("\npush %d",s);
 }
  while(ptr->next!=NULL)
  {
    int v=ptr->vertex;
     if(visited[v]==0)
  {
      parent[v]=s;
      level[v]=level[s]+1;
      dfs(v,p);   //RECURSIVE DFS
  }
   else
    ptr=ptr->next;
  }

 if(ptr->next==NULL)
 {
    int v=ptr->vertex;
     if(visited[v]==0)
   {
      parent[v]=s;
      level[v]=level[s]+1;
      dfs(v,p);
   }

     else
    {
    if(p.last==NULL)
       return;
     int  m=pop(&p);
     printf("\npop %d",m);
      if(p.last==NULL)
       return;
     snode *curr=p.last;
     int i=curr->data;
     dfs(i,p);
   }
 }
 }
 return;
 }
 /*              WITHOUT STACK DFS
 void dfs(int i)
 {
  node* ptr;
  int s=i;
  printf("\nvisit %d",i);
  ptr=head[i];
  visited[i]=1;
  while(ptr!=NULL)
  {
    i=ptr->vertex;
    if(!visited[i])
    parent[i]=s;
    if(!visited[i])
      dfs(i);
   ptr=ptr->next;
 }
 }
 */
 void main()
 {
```

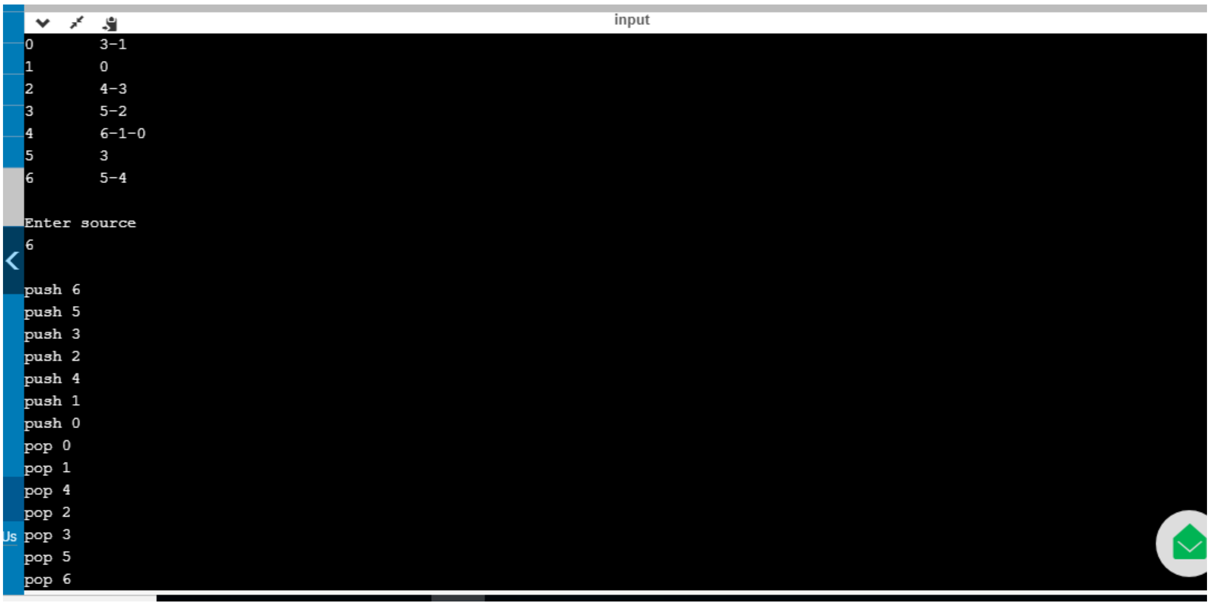
```
int i,j;

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);
for(i-0;i<7;i++)
{
   visited[i]=0;
   parent[i]=-1;
   level[i]=0;
}
  printf("\nThe adjacency list is stored -\n");
  graph(a);                                          //forming adj list
  int x;
stack p;
  p.last=NULL;
 printf("Enter source\n");
 scanf("%d",&x);
 parent[x]=-1;
 dfs(x,p);
 print();
 printf("Enter destination\n");
 scanf("%d",&x);
 getroute(x);
 }
```

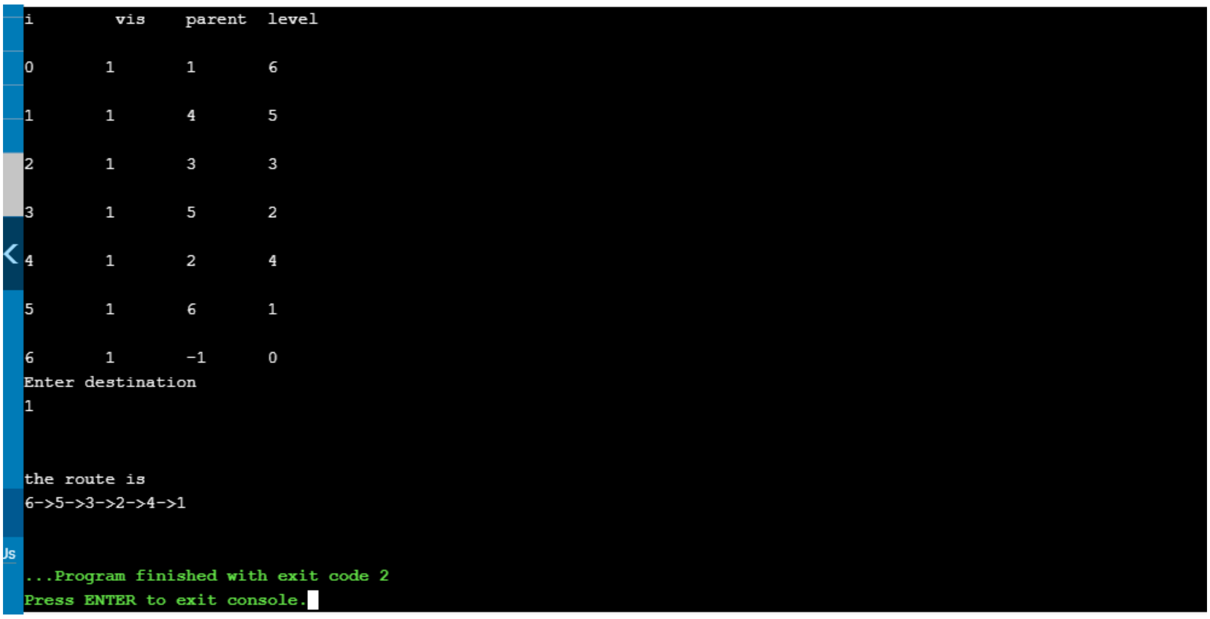
OUTPUT-



```
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 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
```



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

Enter source
6

push 6
push 5
push 3
push 2
push 4
push 1
push 0
pop 0
pop 1
pop 4
pop 2
pop 3
pop 5
pop 6
```



```
i         vis     parent   level

0        1        1        6

1        1        4        5

2        1        3        3

3        1        5        2

4        1        2        4

5        1        6        1

6        1        -1       0
Enter destination
1


the route is
6->5->3->2->4->1


...Program finished with exit code 2
Press ENTER to exit console.
```