**PROGRAM CODE:**

#include<stdio.h>

#include<stdlib.h>

#include "intque.h"

#include "topo.h"

int indegree(graph g,int v)

{

int j,count=0;

for(j=1;j<=g->vertices;j++)

if (g->adj\_mat[j][v]==1)

count++;

return count;

}

void topologicalsort(graph g)

{

int counter=0,i;

int v,w;

queue q=createqueue(g->vertices);

for(i=1;i<=g->vertices;i++)

if(indegree(g,i)==0)

enqueue(q,i);

while(!isempty(q))

{

v=dequeue(q);

printf("%d ",v);

++counter;

for(w=1;w<=g->vertices;w++) //for each vertex

{

if(g->adj\_mat[v][w]==1) //check if its adjacent

{

g->adj\_mat[v][w]-=1;

if(indegree(g,w)==0)

enqueue(q,w);

}

}

}

if(counter!=g->vertices)

printf("\n\nThe graph is cyclic\n");

else

printf("\n\nThe graph is acyclic\n");

}

void main()

{

graph g;

int i,j,v,e;

printf("Enter number of vertices: ");

scanf("%d",&v);

g=create\_graph(v);

printf("\nThe adjacency matrix is\n\n");

display(g);

printf("\nTopological sorting:\n");

topologicalsort(g);

}

**topo header file:**

typedef struct graph\_node \*graph;

struct graph\_node

{

int vertices,edges;

int \*\*adj\_mat;

};

graph create\_graph(int v)

{

int i,n,j,k;

graph g=(graph)malloc(sizeof(struct graph\_node));

if(g==NULL)

{ printf("Out of space\n");

exit(1);

}

g->vertices=v;

int\*\*a=(int\*\*)malloc(sizeof(int\*)\*v+1);

for(i=1;i<v+1;i++)

a[i]=(int\*)malloc(sizeof(int)\*v+1);

for(i=1;i<v+1;i++)

for(j=1;j<v+1;j++)

a[i][j]=0;

printf("Enter the egdes as ViVj (Press -1 to stop):\n");

scanf("%d",&n);

do

{

k=n%10;

j=n/10;

a[j][k]=1;

scanf("%d",&n);

}while(n!=-1);

g->adj\_mat=a;

return g;

}

void display(graph g)

{

int i,j;

int n=g->vertices;

for(i=1;i<n+1;i++)

{ for(j=1;j<n+1;j++)

printf("%d ",g->adj\_mat[i][j]);

printf("\n");

}

}

**intque header file:**

typedef struct queuerecord\* queue;

typedef int elementtype;

struct queuerecord

{

int capacity;

int front;

int rear;

int size;

elementtype \*array;

};

void makeempty(queue q)

{

q->size = 0;

q->front = -1;

q->rear = -1;

}

queue createqueue(int maxelements)

{

int i;

queue q=(queue)malloc(sizeof(struct queuerecord));

if(q==NULL)

printf("Out of space!");

q->array=(elementtype\*)malloc(sizeof(elementtype) \* maxelements);

if(q->array == NULL)

printf("Out of space!");

q->capacity = maxelements;

makeempty(q);

return q;

}

void disposequeue(queue q)

{

if(q!=NULL)

{

free(q->array);

free(q);

}

}

int isempty (queue q)

{

return q->size == 0;

}

int isfull (queue q)

{

return q->size == q->capacity;

}

void enqueue(queue q, elementtype item)

{

if(isfull(q))

printf("Queue is full");

else

{

q->rear++;

q->array[q->rear]=item;

if(q->front==-1)

q->front=0;

++q->size;

}

}

int dequeue(queue q)

{

int data;

if(isempty(q))

{

printf("Queue is empty");

return;

}

else

{

data = q->array[q->front];

if(q->front==q->rear)

{ makeempty(q);

return data; }

else if(q->front+1 == q->capacity)

q->front=0;

else

q->front++;

q->size--;

return data;

}

}

void displayq(queue q)

{

int i;

if(isempty(q))

printf("\nContainer is empty");

else

{ printf("\nThe products in the container are: ");

i=q->front;

while(i!=q->rear)

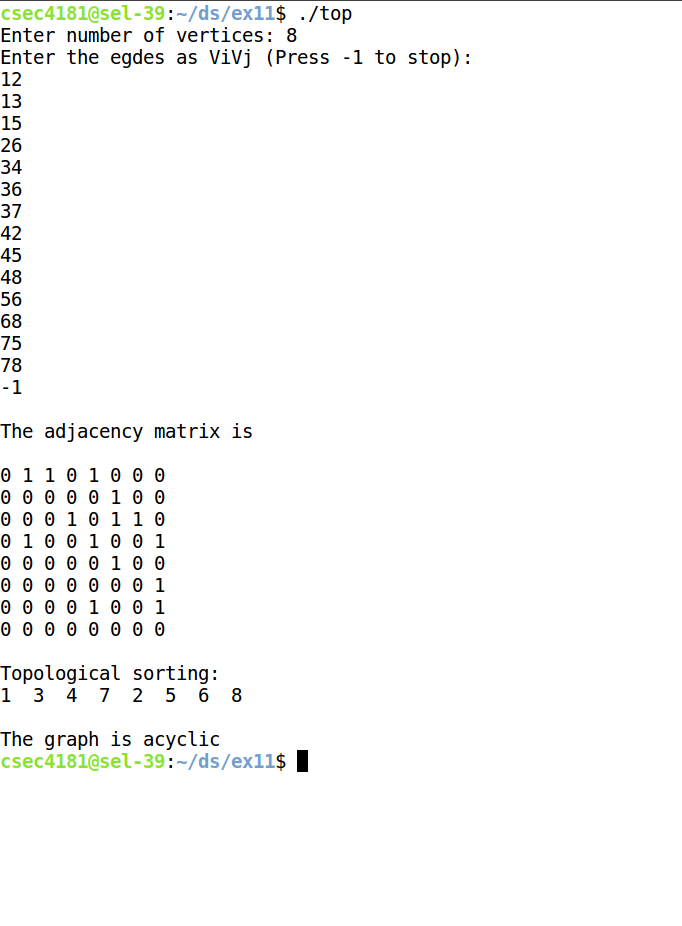
{ printf("\n%d",q->array[i]); i=(i+1)%q->capacity; } //end while

printf("\n%d\n",q->array[q->rear]);

} //end else

}

**OUTPUT:**

****