***ASSIGNMENT-13***

1. ***WAP to construct a Graph using adjacency matrix and implement the following:***

***a. Depth first search.***

#include<stdio.h>

#define SIZE 100

int graph[SIZE][SIZE],n;

int visited[SIZE]={0};

void DFS(int i); //Recursive

void DFS2(int i); //iteartive

void display();

int main()

{

printf("Enter total number of vertices : ");

int ch,x,y;

scanf("%d",&n);

printf("1.Directed Graph\n");

printf("2.Undirected Graph\n");

scanf("%d",&ch);

if(ch==1)

printf("Enter edges as V1 V2(v1-->v2)\n");

else

printf("Enter edges as V1 V2\n");

printf("Enter -1 to stop\n");

while(1)

{

scanf("%d%d",&x,&y);

if(x==-1)

break;

if(ch==1)

graph[x][y]=1;

else

graph[x][y] = graph[y][x] =1;

}

printf("Recursive : ");

DFS(0);

printf("\n");

for(x=0;x<SIZE;x++)

visited[x]=0;

printf("Iterative : ");

DFS2(0);

return 0;

}

void DFS(int i)

{

visited[i]=1;

printf("%d ",i);

int j;

for(j=0;j<n;j++)

if(graph[i][j]==1)

if(visited[j]!=1)

DFS(j);

}

void DFS2(int i)

{

int stack[SIZE];

int top=-1;

stack[++top] = i;

while(top!=-1)

{

int temp = stack[top--];

if(!visited[temp])

printf("%d ",temp);

visited[temp]=1;

int j;

for(j=n-1;j>=0;j--)

if(graph[temp][j]==1)

if(!visited[j])

stack[++top]=j;

}

}

void display()

{

int x,y;

for(x=0;x<n;x++)

{

for(y=0;y<n;y++)

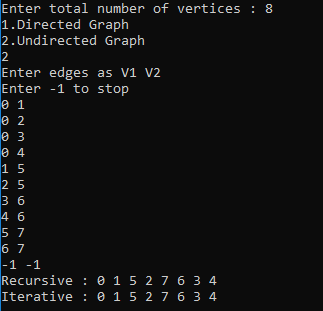
printf("%d ",graph[x][y]);

printf("\n");

}

}

***OUTPUT :***



***b. Breadth first search.***

#include<stdio.h>

#define SIZE 100

int graph[SIZE][SIZE],n;

int visited[SIZE]={0};

void BFS(int i);

void display();

int main()

{

printf("Enter total number of vertices : ");

int ch,x,y;

scanf("%d",&n);

printf("1.Directed Graph\n");

printf("2.Undirected Graph\n");

scanf("%d",&ch);

if(ch==1)

printf("Enter edges as V1 V2(v1-->v2)\n");

else

printf("Enter edges as V1 V2\n");

printf("Enter -1 to stop\n");

while(1)

{

scanf("%d%d",&x,&y);

if(x==-1)

break;

if(ch==1)

graph[x][y]=1;

else

graph[x][y] = graph[y][x] =1;

}

printf("BFS of given graph : ");

BFS(0);

return 0;

}

void BFS(int i)

{

int queue[SIZE];

int front=0,rear=-1;

queue[++rear] = i;

while(rear>=front)

{

int temp = queue[front++];

if(!visited[temp])

printf("%d ",temp);

visited[temp] = 1;

int j;

for(j=0;j<n;j++)

if(graph[temp][j]==1)

queue[++rear] = j;

}

}

void display()

{

int x,y;

for(x=0;x<n;x++){

for(y=0;y<n;y++)

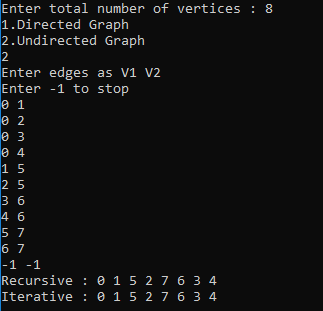
printf("%d ",graph[x][y]);

printf("\n");

}

}

***OUTPUT :***



1. ***WAP to construct a Graph using adjacency list representation and implement the following:  
   a. Create an edge between two nodes.  
   b. Remove an edge between two nodes.  
   c. Degree of a particular node.  
   d. Create a new node.  
   e. Remove an existing node.***

#include<stdio.h>

#include<stdlib.h>

struct graphnode{

int info;

struct graphnode \*next;

};

typedef struct graphnode Graph;

void create\_edge(Graph \*\*graph,int ch,int x,int y);

void delete\_edge(Graph \*\*graph,int ch,int x,int y);

void find\_degree(Graph \*\*graph,int x,int n);

Graph \*\*create\_node(Graph \*\*graph,int \*n,int ch);

Graph \*\*delete\_node(Graph \*\*graph,int x,int ch,int \*n);

void display(Graph \*\*graph,int n);

int main()

{

printf("Enter total number of vertices : ");

int ch,x,y,n,temp;

Graph \*p;

scanf("%d",&n);

Graph \*\*graph = (Graph \*\*)malloc(n\*sizeof(Graph \*));

for(x=0;x<n;x++)

graph[x]=NULL;

printf("1.Directed Graph\n");

printf("2.Undirected Graph\n");

scanf("%d",&ch);

if(ch==1)

printf("Enter edges as V1 V2(v1-->v2)\n");

else

printf("Enter edges as V1 V2\n");

printf("Enter -1 to stop\n");

while(1)

{

scanf("%d%d",&x,&y);

if(x==-1)

break;

create\_edge(graph,ch,x,y);

}

printf("1. Create an edge between two nodes.\n");

printf("2.Remove an edge between two nodes.\n");

printf("3. Degree of a particular node.\n");

printf("4. Create a new node.\n");

printf("5.Remove an existing node.\n");

printf("6.Display adjancency list.\n");

printf("7.Exit\n");

while(1)

{

scanf("%d",&temp);

switch(temp)

{

case 1:

printf("Enter vertices having edge :");

scanf("%d%d",&x,&y);

create\_edge(graph,ch,x,y);

break;

case 2:

printf("Enter vertices having edge :");

scanf("%d%d",&x,&y);

delete\_edge(graph,ch,x,y);

break;

case 3:

printf("Enter Node number :");

scanf("%d",&x);

find\_degree(graph,x,n);

break;

case 4:

graph = create\_node(graph,&n,ch);

break;

case 5:

printf("Enter node which is to be deleted :");

scanf("%d",&x);

graph = delete\_node(graph,x,ch,&n);

break;

case 6:

display(graph,n);

break;

case 7:

exit(1);

}

}

return 0;

}

void create\_edge(Graph \*\*graph,int ch,int x,int y)

{

Graph \*temp = (Graph \*)malloc(sizeof(Graph));

temp->info = y;

temp->next = graph[x];

graph[x] = temp;

if(ch==2)

{

Graph \*temp2 = (Graph \* )malloc(sizeof(Graph));

temp2->info = x;

temp->next = graph[y];

graph[y] = temp2;

}

}

void delete\_edge(Graph \*\*graph,int ch,int x,int y)

{

Graph \*p,\*q;

p = q = graph[x];

while(p!=NULL && p->info!=y)

{

q = p;

p = p->next;

}

if(p!=NULL)

{

if(q==p)

graph[x] = p->next;

else

q->next = p->next;

free(p);

}

if(ch==2)

{

p = q = graph[y];

while(p!=NULL && p->info!=x)

{

q = p;

p = p->next;

}

if(p!=NULL)

{

if(q==p)

graph[y] = p->next;

else

q->next = p->next;

free(p);

}

}

}

void find\_degree(Graph \*\*graph,int x,int n)

{

int in=0,out=0;

Graph \*p;

int i;

for(i=0;i<n;i++)

{

p = graph[i];

while(p!=NULL)

{

if(i==x)

out++;

if(p->info==x)

in++;

p = p->next;

}

}

printf("Out-Degree : %d\nIn Degree : %d\n",out,in);

}

Graph \*\*create\_node(Graph \*\*graph,int \*n,int ch)

{

int x,y;

(\*n)++;

graph = (Graph \*\*)realloc(graph,(\*n)\*sizeof(Graph \*));

graph[\*n-1] = NULL;

printf("New node %d is created\n",\*n-1);

printf("Enter edges connected to node %d (-1 to stop)\n",\*n-1);

while(1)

{

scanf("%d%d",&x,&y);

if(x==-1)

break;

create\_edge(graph,ch,x,y);

}

return graph;

}

Graph \*\*delete\_node(Graph \*\*graph,int x,int ch,int \*n)

{

int i;

for(i=0;i<\*n;i++)

if(i!=x)

delete\_edge(graph,ch,i,x);

free(graph[x]);

for(i=x;i<\*n-1;i++)

graph[i] = graph[i+1];

(\*n)--;

return graph;

}

void display(Graph \*\*graph,int n)

{

int x;

Graph \*p;

for(x=0;x<n;x++)

{

p = graph[x];

printf("%d ",x);

while(p!=NULL)

{

printf("--> %d ",p->info);

p = p->next;

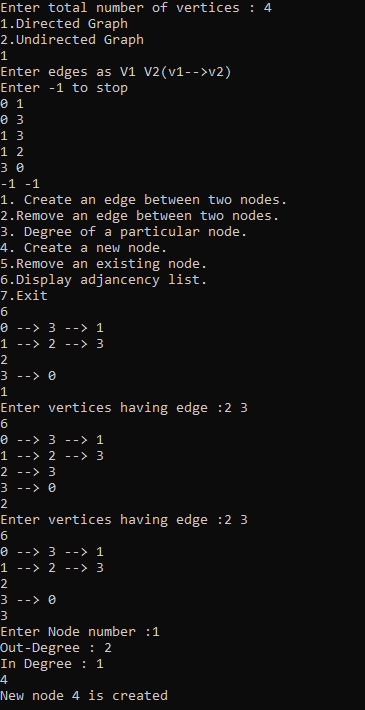
}

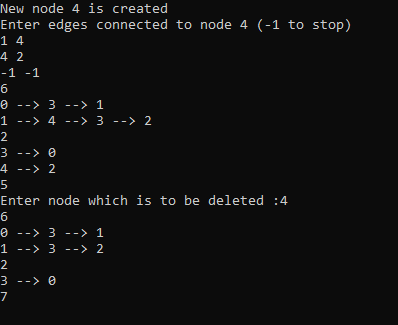
printf("\n");

}

}

***OUTPUT :***





***/\*The end\*/***