Module 5 Graphs

73. Program for Breadth First Search #include<iostream.h> #include<conio.h> #include<stdlib.h> int cost[10][10], i, j, k, n, queue[10], front, rear, v, visit[10], visited[10]; void main() { int m; clrscr(); cout <<"enter no of vertices"; cin >> n; cout <<"enter no of edges"; cin >> m; cout <<"\n EDGES \n"; for(k=1;k<=m;k++) { cin >>i>>j; cost[i][j]=1; } cout <<"enter initial vertex"; cin >>v; cout <<"Visited vertices\n"; cout << v; visited[v]=1; k=1; while(k<n)  
{ for(j=1;j<=n;j++) if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1) { visit[j]=1; queue[rear++]=j; } v=queue[front++]; cout<<v << " ";  
k++; visit[v]=0; visited[v]=1; }  
getch(); }

74. Program for Depth First Search #include<iostream.h> #include<conio.h> #include<stdlib.h> int cost[10][10],i,j,k,n,stack[10],top,v,visit[10],visited[10]; void main() { int m; cout <<"enter no of vertices"; cin >> n; cout <<"enter no of edges"; cin >> m; cout <<"\n EDGES \n";  
for(k=1;k<=m;k++) { cin>>i>>j; cost[i][j]=1; } cout <<"enter initial vertex"; cin >>v; cout <<"ORDER OF VISITED VERTICES"; cout << v <<" "; visited[v]=1; k=1; while(k<n) { for(j=n;j>=1;j--) if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1) { visit[j]=1; stack [top]=j; top++; } v= stack[--top]; cout<<v << " "; k++; visit[v]=0; visited[v]=1; } getch(); }

91. Program for the implementation of Breadth First Search (BFS) for a given graph  
#include<iostream> #include<conio.h> #include<stdlib.h> using namespace std; int cost[10][10],i,j,k,n,qu[10],front,rare,v,visit[10],visited[10]; main() { int m; cout <<"enterno of vertices"; cin >> n; cout <<"ente no of edges"; cin >> m; cout <<"\nEDGES \n"; for(k=1;k<=m;k++) { cin >>i>>j; cost[i][j]=1; } cout <<"enter initial vertex";  
cin >>v; cout <<"Visitied vertices\n"; cout << v; visited[v]=1; k=1; while(k<n) { for(j=1;j<=n;j++) if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1) {  
visit[j]=1; qu[rare++]=j; } v=qu[front++]; cout<<v << " "; k++; visit[v]=0; visited[v]=1;  
} }

92. Program for the implementation of Depth-first search (DFS) for a given graph  
#include<iostream> #include<conio.h> #include<stdlib.h> using namespace std; int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10];  
main() { int m; cout <<"enterno of vertices"; cin >> n; cout <<"ente no of edges"; cin >> m; cout <<"\nEDGES \n"; for(k=1;k<=m;k++) { cin >>i>>j; cost[i][j]=1; } cout <<"enter initial vertex"; cin >>v; cout <<"ORDER OF VISITED VERTICES"; cout << v <<" "; visited[v]=1; k=1; while(k<n) { for(j=n;j>=1;j--) if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1){  
visit[j]=1; stk[top]=j; top++; } v=stk[--top]; cout<<v << " "; k++; visit[v]=0; visited[v]=1; } }

93. Code to get Transpose of a Graph  
Graph Graph::getTranspose() { Graph g(V); for (int v = 0; v < V; v++) { list<int>::iterator i; for(i = adj[v].begin(); i != adj[v].end(); ++i) { g.adj[\*i].push\_back(v); } } return g; }

94. Code for Deletion of element from the Doubly Linked List void double\_llist::delete\_element(int value) { struct node \*tmp, \*q; /\*first element deletion\*/ if (start->info == value) { tmp = start; start = start->next; start->prev = NULL; cout<<"Element Deleted"<<endl; free(tmp); return; } q = start; while (q->next->next != NULL) { /\*Element deleted in between\*/ if (q->next->info == value) { tmp = q->next; q->next = tmp->next; tmp->next->prev = q; cout<<"Element Deleted"<<endl; free(tmp);  
return; } q = q->next; } /\*last element deleted\*/ if (q->next->info == value) { tmp = q->next; free(tmp); q->next = NULL; cout<<"Element Deleted"<<endl; return; } cout<<"Element "<<value<<" not found"<<endl; }

95. Code to insert at a particular position in a Doubly Linked List void double\_llist::add\_after(int value, int pos) { if (start == NULL) { cout<<"First Create the list."<<endl; return; } struct node \*tmp, \*q; int i; q = start; for (i = 0;i < pos - 1;i++) { q = q->next; if (q == NULL) { cout<<"There are less than "; cout<<pos<<" elements."<<endl; return; } } tmp = new(struct node); tmp->info = value; if (q->next == NULL) { q->next = tmp; tmp->next = NULL; tmp->prev = q; } else { tmp->next = q->next; tmp->next->prev = tmp; q->next = tmp;  
tmp->prev = q; } cout<<"Element Inserted"<<endl; }

96. Program to find MST(Minimal Spanning Tree) using Prim's Algorithm #include <iostream> #include <conio.h> using namespace std; struct node { int fr, to, cost; }p[6]; int c = 0, temp1 = 0, temp = 0; void prims(int \*a, int b[][7], int i, int j) { a[i] = 1; while (c < 6) { int min = 999; for (int i = 0; i < 7; i++) { if (a[i] == 1) { for (int j = 0; j < 7; ) { if (b[i][j] >= min || b[i][j] == 0) j++; else if (b[i][j] < min) { min = b[i][j]; temp = i; temp1 = j; } } } } a[temp1] = 1; p[c].fr = temp; p[c].to = temp1; p[c].cost = min; c++; b[temp][temp1] = b[temp1][temp]=1000; } for (int k = 0; k < 6; k++) { cout<<"source node:"<<p[k].fr<<endl; cout<<"destination node:"<<p[k].to<<endl;  
cout<<"weight of node"<<p[k].cost<<endl; } } int main() { int a[7]; for (int i = 0; i < 7; i++) a[i] = 0; int b[7][7]; for (int i = 0; i < 7; i++) { cout<<"enter values for "<<(i+1)<<" row"<<endl; for (int j = 0; j < 7; j++) cin>>b[i][j]; } prims(a, b, 0, 0); getch(); }

97. Program to Implement Floyd-Warshall Algorithm #include <iostream> #include <conio.h> using namespace std; void floyds(int b[][7]) { int i, j, k; for (k = 0; k < 7; k++) { for (i = 0; i < 7; i++) { for (j = 0; j < 7; j++) { if ((b[i][k] \* b[k][j] != 0) && (i != j)) { if ((b[i][k] + b[k][j] < b[i][j]) || (b[i][j] == 0)) b[i][j] = b[i][k] + b[k][j]; } } } } for (i = 0; i < 7; i++) { cout<<"\nMinimum Cost With Respect to Node:"<<i<<endl; for (j = 0; j < 7; j++) cout<<b[i][j]<<"\t"; } } int main() { int b[7][7]; cout<<"ENTER VALUES OF ADJACENCY MATRIX\n\n";  
for (int i = 0; i < 7; i++) { cout<<"enter values for "<<(i+1)<<" row"<<endl; for (int j = 0; j < 7; j++) cin>>b[i][j]; } floyds(b); getch(); }

98. Code for graph using Adjacency List  
void addReverseEdge(int src,int dest) { np1 = new adj\_list; np1->dest = src; np1->next = NULL; if (array[dest].ptr == NULL) { array[dest].ptr = np1; q = array[dest].ptr; q->next = NULL; } else { q = array[dest].ptr; while (q->next != NULL) { q = q->next; q->next = np1; } } void addEdge(int src,int dest) { np<!--more--> = new adj\_list; np->dest = dest; np->next = NULL; if (array[src].ptr == NULL) { array[src].ptr = np; p = array[src].ptr; p->next = NULL; } else { p = array[src].ptr; while (p->next != NULL) p = p->next; p->next = np; } addReverseEdge(src,dest); }