Q. 1. Write a menu driven program to perform following operations on singly linked list: Create,

Insert, Delete, and Display

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* head = NULL;

void insert(int x) {

Node\* temp = new Node();

temp->data = x;

temp->next = head;

head = temp;

}

void Delete(int n) {

Node\* temp1 = head;

if(n == 1) {

head = temp1->next;

delete temp1;

return;

}

for(int i=0; i<n-2; i++) {

temp1 = temp1->next;

}

Node\* temp2 = temp1->next;

temp1->next = temp2->next;

delete temp2;

}

void display() {

Node\* temp = head;

while(temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

int main() {

int choice, x, n;

while(1) {

cout << "1. Insert" << endl;

cout << "2. Delete" << endl;

cout << "3. Display" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch(choice) {

case 1: cout << "Enter the element: ";

cin >> x;

insert(x);

break;

case 2: cout << "Enter the element you want to delete: ";

cin >> n;

Delete(n);

break;

case 3: display();

break;

case 4: exit(0);

default: cout << "Invalid Input" << endl;

}

}

return 0;

}

Q. 2. Implement minimum cost spanning tree algorithm.

#include <iostream>

#include <vector>

#include <utility>

#include <algorithm>

using namespace std;

const int MAX = 1000;

int id[MAX], nodes, edges; //array id is use for check the parent of vertex;

pair <long long, pair<int, int> > p[MAX];

//initialise the parent array id[]

void init()

{

for(int i = 0;i < MAX;++i)

id[i] = i;

}

int root(int x)

{

while(id[x] != x) //if x is not itself parent then update its parent

{

id[x] = id[id[x]];

x = id[x];

}

return x; //return the parent

}

//function for union

void union1(int x, int y)

{

int p = root(x);

int q = root(y);

id[p] = id[q];

}

//function to find out the edges in minimum spanning tree and its cost

long long kruskal(pair<long long, pair<int, int> > p[])

{

int x, y;

long long cost, minimumCost = 0;

for(int i = 0;i < edges;++i)

{

x = p[i].second.first;

y = p[i].second.second;

cost = p[i].first;

if(root(x) != root(y))

{

minimumCost += cost;

cout<<x<<" ----> "<<y<<" :"<<p[i].first<<endl;//print the edges contain in

spanning tree

union1(x, y);

}

}

return minimumCost;

}

int main()

{

int x, y;

long long weight, cost, minimumCost;

init();

cout <<"Enter Nodes and edges"<<endl;

cin >> nodes >> edges;

//enter the vertex and cost of edges

for(int i = 0;i < edges;++i)

{

cout<<"Enter the value of X, Y and edges"<<endl;

cin >> x >> y >> weight;

p[i] = make\_pair(weight, make\_pair(x, y));

}

//sort the edges according to their cost

sort(p, p + edges);

minimumCost = kruskal(p);

cout <<"Minimum cost is "<< minimumCost << endl;

return 0;

}