CSCE 221 Cover Page
Programming Assignment #6
Due May 5 at 10:00AM to CSNet

Jessica Fang UIN: 224003796

User Name: sndnzki E-mail address: jessicafang@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources					
People	Katherine Click	Ryan Yantz			
Web pages (provide URL)					
Printed material					
Other Sources					

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Jessica Fang Date 04/16/2015

a) This program is able to take in characters from which it creates graphs from an input file. The purpose of this assignment is to utilize our knowledge of the graph and disjoint set data structures and the MST algorithm to create an efficient program which can create a graph and find the minimum spanning tree.

b) void buildGraph() creates an empty adjacency list and edge list

void insertEdge(int i, int j, double w) creates an edge between two nodes if the edge does not exist.

double getWeight(int i, int j) traverses through the edge list to find the edge with an i and j values that match the given i and j values and returns the weight of the edge or 0 if not found.

void sortEdge() uses STL's vector's sort and a less than operator to sort the edge list

double MSTAlgo() implementation of Kruskal's algorithm. creates a set for every node. sorts the edges in the edge list. unions the smallest sets to create a minimum spanning tree. counts and returns the total weight of the minimum spanning tree.

c) Runtime analysis:

void buildGraph() O(n)

void insertEdge(int i, int j, double w) O(log n)

double getWeight(int i, int j) O(log n)

void sortEdge() O(n log n)

double MSTAlgo() O(n log n)

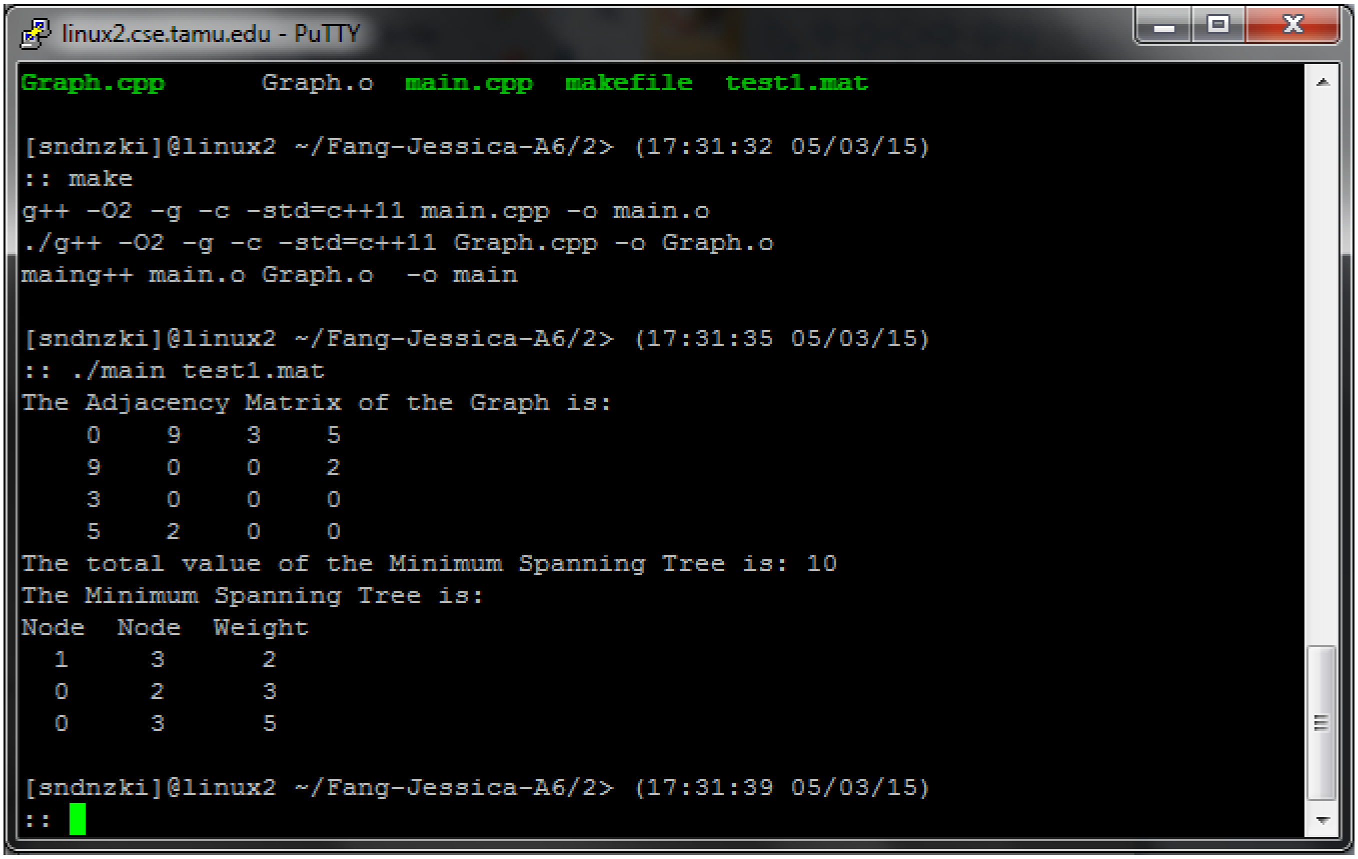
d) This program is compiled using a makefile (type make) and run using ./main.

e) Union does not check if elements in the smaller already exist in larger.

f) DListNode is templated.

g) Tests in the test program run smoothly.

Testing results:



```
Graph.cpp       Graph.o  main.cpp  makefile  test1.mat

[sndnzki]@linux2 ~/Fang-Jessica-A6/2> (17:31:32 05/03/15)
:: make
g++ -O2 -g -c -std=c++11 main.cpp -o main.o
./g++ -O2 -g -c -std=c++11 Graph.cpp -o Graph.o
maing++ main.o Graph.o  -o main

[sndnzki]@linux2 ~/Fang-Jessica-A6/2> (17:31:35 05/03/15)
:: ./main test1.mat
The Adjacency Matrix of the Graph is:
    0    9    3    5
    9    0    0    2
    3    0    0    0
    5    2    0    0
The total value of the Minimum Spanning Tree is: 10
The Minimum Spanning Tree is:
Node  Node  Weight
  1     3      2
  0     2      3
  0     3      5

[sndnzki]@linux2 ~/Fang-Jessica-A6/2> (17:31:39 05/03/15)
::
```