Lab 5 Approach Document

# Minimum Cost Spanning Tree (MST)

# Assignment Objective

# Prim's technique for building Minimum Spanning Trees (MSTs) should be implemented by the system, which should also write each MST to a different file and read and write graphs in a text file format.

# Assignment Requirements

Your system should have the ability to

* **Managing Files:**
  + Create and put into use routines that allow you to read and write graphs to text files.
  + Make sure the graph file formats for the input and output are consistent.
* **Prim's MST Algorithm:**
  + For the purpose of creating Minimum Spanning Trees (MSTs), apply Prim's algorithm.
  + Change the starting vertex to generate three different MSTs.
* **Handling Output:**
  + Write a different file for every MST.

# Approach

* Review all files (documents, code, helps, etc.) provided as part of the assignment
* Select a data structure and implementation that could satisfy this assignment:
  + Design a format for storing graphs in a file (must be a text file readable with a text editor)
  + Then implement two functions:
    - one to read a graph from a file
    - the other to write a graph to a file (your input and output graph files should be in the same format).
  + Test your functions by implementing acomplete MST program that reads an undirected graph from a file
    - constructs the MST using Prim’s algorithm
    - writes to a second file the graph representing the MST
  + Since we are going to build three MSTs there should be three output files (one for each MST)
    - For this assignment we must incorporate smart pointers to get full credit
  + Build three (3) MSTs by changing the starting vertex. Use the following three vertexes for the starting vertex: 0, 4, 2
  + void start\_operation(int current\_vertex)
    - Initalize the 2D matrix of all zeros
    - Create a priority queue using a vector of integer pairs (We need to have the functionality of a priority queue but full access to all the items of the vector)
    - Loop until all vertices are visited
      * Insert all values into the priority queue before we decide which one to take
        + If we have a value that is not 0 and we have not visited it yet then we add it to the queue
        + Insert the value and the index
        + Organize the queue
      * Get the top value from the queue
      * Print out the edge
      * Add the value into the new matrix (2 places because of the way it is mirrored)
      * Add the index to the ignore list and remove all elements from the queue that have that index

# Build Log

11/15/23 – Today I finished going over all the assignment documentation and other files and have created my approach document up to this point. I created the project folder and met with Cameron Kauffman to work together on the lab. Created MST.h and main.cpp files

11/27/23 – Met with Cameron and finished the first draft of the program. Meeting with the tutor tomorrow for smart pointer implementation and bug fixes.

11/28/23 – Met with the tutor to fix the smart pointer issue and finish up the implementation. Packaged up files and submitted today as well