* **Miscellaneous Notes**
  + I did not utilize any provided files for this lab.
  + I worked with Bryce Keen on this lab.
* **smartMST.h**
  + **Private Data Members**
    - Only three variables will need to be used throughout the header – the rest can be stored within functions as their scope will not extend past their parent functions. The three private data members are two smart pointer 2D arrays of type int, to store the provided undirected graph. They are created as smart pointers so that it will be capable of handling undirected graphs with any number of vertices to calculate the minimum spanning tree. The last data member is an int to store the number of vertices found in the file.
  + **Default Constructor**
    - The default constructor is quite complex, mostly because it must correctly initialize the dynamically allocated smart pointer arrays, read in the undirected graph from the file, and store its contents in both arrays. Passed into the constructor will be the name of the file which contains the undirected graph. Variables must be created to store each edge weight read in from the file and to store each line input for indexing of vertices. A filestream object will need to be created for file operations. First, the function will count the number of lines to determine the number of vertices. Next, it will properly initialize the smart pointers, dynamically sizing them based on the number of vertices found in the undirected graph – all values in each 2D array will be initialized to zero. Finally, the arrays will be populated with the values from the file, and the file will be closed.
  + **Destructor**
    - Since smart pointers are utilized for the dynamic 2D arrays, nothing is required for the destructor, as the pointers will be automatically deleted as soon as they leave the scope of their operation.
  + **getVerts/getEdges**
    - Both functions will return the numVerts variable, but getEdges will return numVerts minus one, since the number of edges will always be one less than the number of vertices on a minimum spanning tree.
  + **prim**
    - This function will be based on Prim’s algorithm and will be used to populate graph2 with the edges necessary for the minimum spanning tree. The function will take a passed in int, which will be the starting vert for calculating the MST. Another smart pointer array will be created, the size of the number of found vertices, and will be used to track which vertices have been checked to find the minimum edge. A series of for loops and if statements will be used to start at the passed in vertex and iterate over the edges in the undirected graph, finding the minimum edge connecting each vertex, limited to the desired number of edges (vertices – 1). Values will be assigned to graph2 for printing.
  + **printGraph**
    - This function will both print graph2 and write its contents to a file (name passed in).
* **Driver CPP**
  + The driver will call functions from the class, first printing the number of vertices, edges, and the staring matrix, then executing three MST searches starting at vertices 0, 4, and 2.

