Project 7 (C++) : Given a connected undirected graph, G, using the Prim's algorithm to construct the Minimum Spanning Tree of G.
(This is a easy project, start early to get extra credit!)
Prim's algorithm uses two sets, setA and setB to construct a MST of G. Initially, setA contains only one node (can be any node in G) and setB contains the rest of nodes. You will run your program four (4) times, using 1, 4, 8, 12 on each run as the initial node in setA.

Include in your hard copy:
- cover page
- draw the cost matrix of the graph
- draw the illustration of the MST construction process for setA = 4 and setA = 8
- source code
- print MSTfile for setA = 1
- print deBugFile for setA = 1
- print MSTfile for setA = 4
- print MSTfile for setA = 8
- print MSTfile for setA = 12

*** You do not need to print deBugFile for 4, 8, and 12.

****

Language: C++
Project points: 10pts
Due Date: Soft copy (*.zip) and hard copies (*.pdf):

10/10 on time: 4/28/2021 Wednesday before midnight
+1 early submission: 4/25/2021 Sunday before midnight
-1 for 1 day late: 4/29/2021 Thursday before midnight
-2 for 2 days late: 4/30/2021 Friday before midnight
-10/10: 4/30/2021 Friday after midnight
-5/10: does not pass compilation
0/10: program produces no output
0/10: did not submit hard copy.

*** Follow "Project Submission Requirement" to submit your project.

****

I. Inputs: There are two inputs to the program
a) inFile (argv [1]): A text file contains a connected undirected graph, as a list of edges with costs, <n_i, n_j, c>.
The first text line is the number of nodes in G, follows by a list of triplets, <n_i, n_j, cost>; $n_i > 0$ and $n_j > 0$.
Please note that an undirected edge includes two directed edges; therefore, the cost matrix is symmetry.
For example:
```
5        // there are 5 nodes in the graph
1  5 10  // an undirected edge: i.e., an edge <1, 5, 10> and an edge <5, 1, 10>
2  3  5  // an undirected edge: i.e.,  an edge <2, 3, 5> and an edge <3, 2, 5>
1  2  20
3  5  2
:
```
b) nodeInSetA (argv[2]) // user select a node to be in setA.

****

II. Output: MSTfile (argv [3]): The result of Prim's MST in text file format.

Your program output should look like the following:
```
*** Prim's MST of the input graph, G is: ***
        5
        2  3  5
        3  5  2
        :
*** The total cost of  the Prim's MST is: whatever
```
b) debugFile (args[2]) : For all other outputs.

****
III. Data structure:
****

- An uEdge class
    - (int) Ni // an integer 1 to N
    - (int) Nj // an integer 1 to N
    - (int) cost // a positive integer > 0
    - (uEdge *) next

  Methods:
    - constructor (…) // to create a new uEdge node with (Ni, Nj , cost, null)
    - printEdge (edge, deBugFile) // print edge info to deBugFile

- A PrimMST class
  - (int) numNodes //number of nodes in G.
  - (int) nodeInSetA // get from argv[2]
  - (int) whichSet [] // a 1-D integer array, size of numNodes +1, dynamically allocated.
    // to indicate which set each node belongs to (1 for setA or 2 for setB).

  - (uEdge *) edgelistHead // pointer to a linked list of uEdge of the input graph edges in ascending order;
    // so that to find a min cost edge would be simple, i.e., at the front of the list, with O(1).
    // Initially, it points to a dummy node <0, 0, 0, null> when created.

  - (uEdge *) MSTlistHead // pointer to a linked list of uEdge nodes in Prim's MST.
    // Initially, it points to a dummy node <0, 0, 0, null>
  - totalMSTCost (integer) // initially set to zero

  Methods:

  - listInsert (edge) // inserts edge into the list of edgelistHead in ascending order w.r.t. edge cost.
    // Re-use code in your previous projects.

  - (uEdge *) removeEdge (…) // The method searches edge nodes in edgelistHead linked list
    // to find the first edge, e, where: a) whichSet[e->Ni] != whichSet[e->Nj]
    // && b) either whichSet[e->Ni] == 1 or whichSet[e->Nj] == 1.
    // Then, remove edge node e from the list and returns e.
    // Re-use code in your previous projects.

  - addEdge (edge) // add edge on the top of MSTlistHead after dummy. NOT sorted!
    // Re-use code in your previous projects.

  - printSet (…) // print the whichSet array to deBugFile. Re-use code in your previous projects.

  - printEdgeList (…) // print nodes in the list point by edgelistHead to deBugFile with the following
    // edgelistHead → <0, 0, 0, N1> → <N1, N1, edgeCost, N2> → <N2, N2, edgeCost, N3> …
    // Re-use code in your previous projects.

  - printMSTList (outFile) // print nodes in the list point by MSTlistHead to outFile with the following
    // MSTlistHead → <0, 0, 0, N1> → <N1, N1, edgeCost, N2> → <N2, N2, edgeCost, N3> …
    // Re-use code in your previous projects.

  - (bool) setBisEmpty(…) // returns true if whichSet are all 1, otherwise returns false

  - updateMST (newEdge) // see algorithm below

*** You may add methods or variable is necessary.

```
************************************
IV. main (…)
************************************
Step 0:  inFile, MSTfile, deBugFile ← open
         numNodes ← get from inFile
         nodeInSetA  ← get from argv[2]
         whichSet[] ← dynamically allocated, size of numNodes+1, and initialize to 2
         whichSet[nodeInSetA] ← 1
         printSet (…)
         edgelistHead ← get a dummy node <0, 0, 0, null> for it to point to.
         MSTlistHead ← get a dummy node <0, 0, 0, null> for it to point to.
         totalMSTCost ← 0

Step 1: Ni, Nj, edgeCost ← read from inFile
         newEdge ← get a new uEdge node with (Ni, Nj, edgeCost, null)
         listInsert (newEdge)

Step 2: printEdgeList (…) // print to deBugFile

Step 3: repeat step 1 to step 2 until the inFile is empty.

Step 4: nextEdge ← removeEdge (…)

Step 5: printEdge (newEdge, deBugFile)

Step 6: updateMST (newEdge)

Step 7: printSet (deBugFile)

Step 8: printEdgeList (deBugFile)
         printMSTList (deBugFile)

Step 9: repeat step 4 – step 8 until  setBisEmpty(…)  // whichSet are all 1's

Step 10: print "*** Prim's MST of the input graph, G is: ***" to MSTfile
          print numNodes to MST file
          printMSTList (MSTfile)
          print " *** MST total cost = " MSTtotalCost to MSTfile

Step 11: close all files.

************************************
V. updateMST (newEdge)
************************************
Step 1: addEdge (newEdge) // adding newEdge to MST list

Step 2: totalMSTCost += newEdge ->cost

Step 3:  if whichSet [newEdge -> Ni] == 1 // Ni is in setA,
             whichSet[newEdge -> Nj] ← 1 // move Nj from setB to setA
       else
             whichSet[newEdge -> Ni] ← 1 // move Ni from setB to setA
```