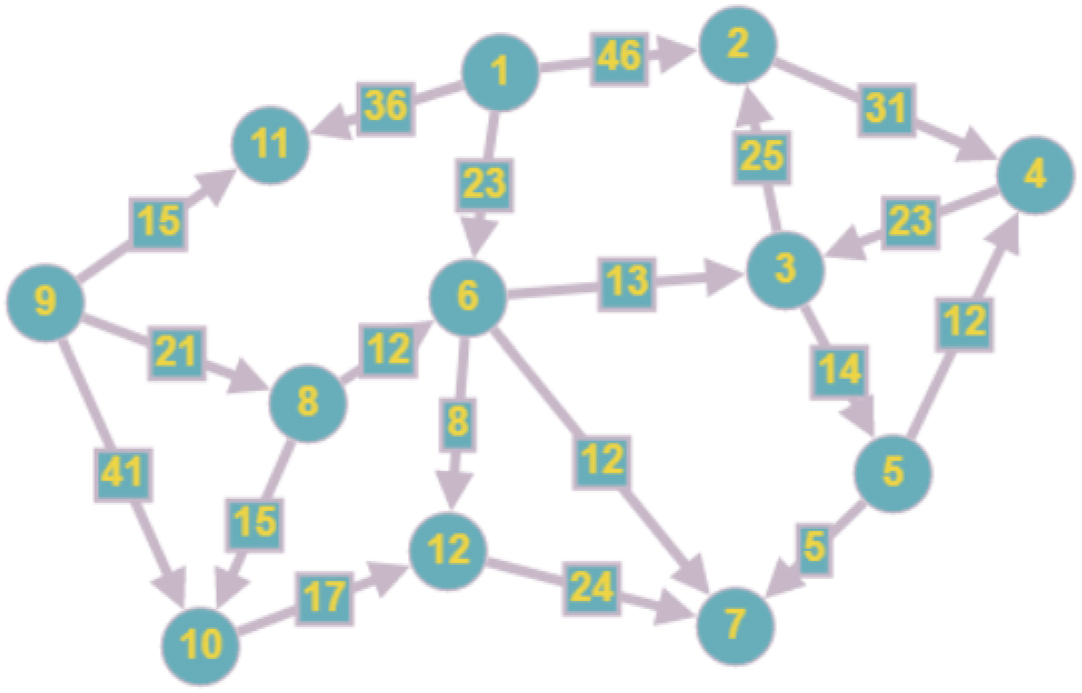
323-34 Project 7: Prim MST **C++**

Student: Seratul Ambia

Project Due Date: 4/28/2021

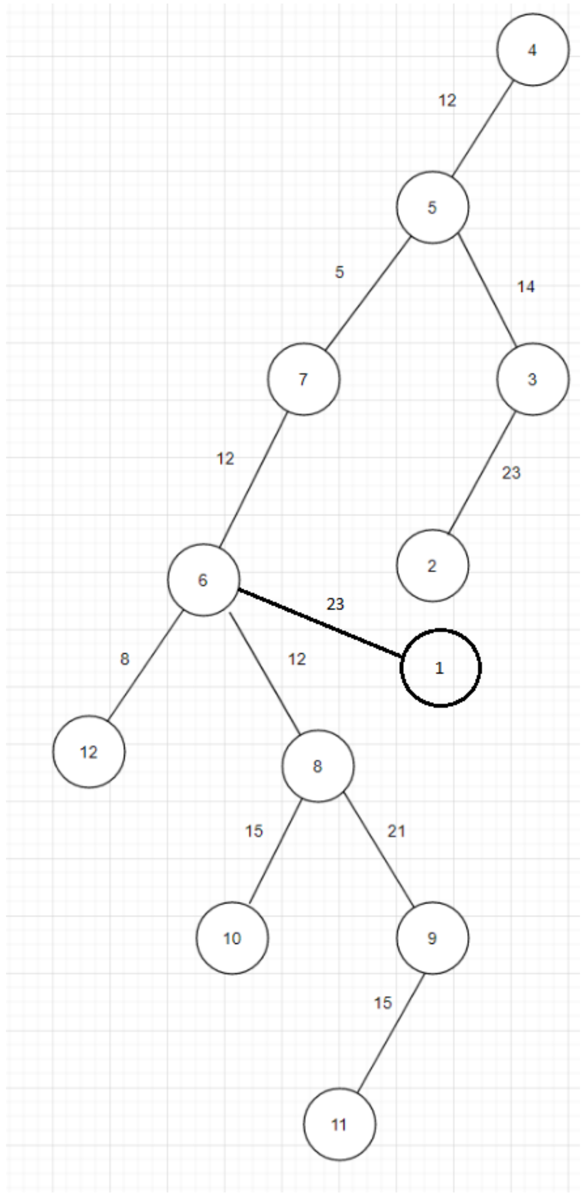
<u>Adjacency Graph:</u>



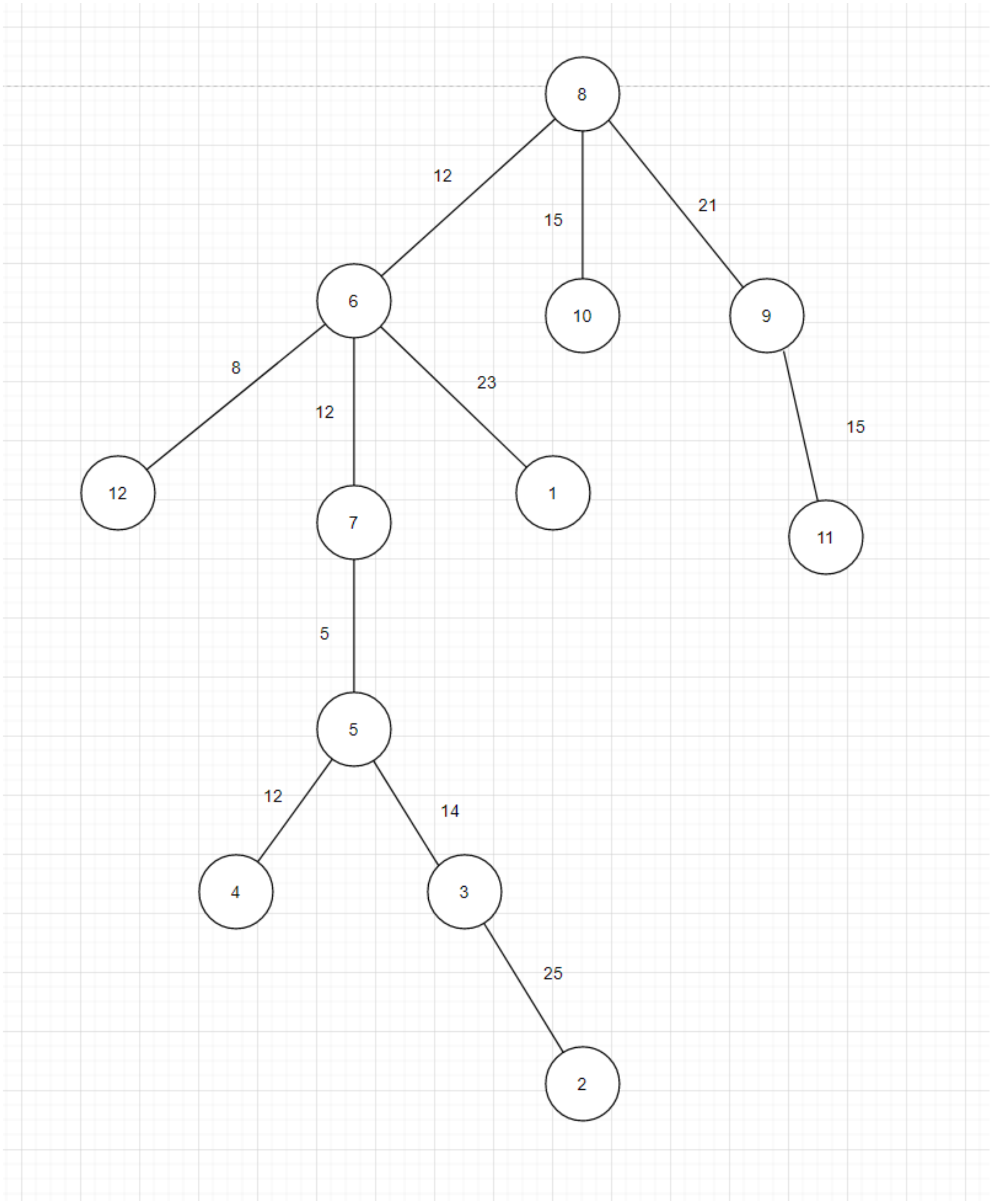
	1	2	3	4	5	6	7	8	9	10	11	12
1	9999	46	9999	9999	9999	23	9999	9999	9999	9999	36	9999
2	9999	9999	9999	31	9999	9999	9999	9999	9999	9999	9999	9999
3	9999	25	9999	9999	14	9999	9999	9999	9999	9999	9999	9999
4	9999	9999	23	9999	9999	9999	9999	9999	9999	9999	9999	9999
5	9999	9999	9999	12	9999	9999	5	9999	9999	9999	9999	9999
6	9999	9999	9999	13	9999	9999	12	9999	9999	9999	9999	8
7	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999
8	9999	9999	9999	9999	9999	12	9999	9999	9999	15	9999	9999
9	9999	9999	9999	9999	9999	9999	9999	21	9999	9999	15	9999
10	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	17

11	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999
12	9999	9999	9999	9999	9999	9999	24	9999	9999	9999	9999	9999

Set A=4 Construction Process:



Set A=8 Construction Process:



Source Code:

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;
```

```
class uEdge {
public:
   int Ni, Nj, cost;
   uEdge* next;

   uEdge(int i, int j, int c) {
      this->Ni = i;
      this->Nj = j;
      this->cost = c;
      this->next = NULL;
   }

   void printEdge(uEdge *edge, ofstream &debugFile) {
      if (edge->next != NULL)
         debugFile << "<" << edge->Ni << ", " << edge->Nj << ", " << edge->cost << ", " <<
edge->next->Ni << ">";
      else
         debugFile << "<" << edge->Ni << ", " << edge->Nj << ", " << edge->cost << ", " <<
"NULL" << ">";
   }
};

class PrimMST {
public:
   int numNodes;
   int nodeInSetA;
   int* whichSet;
   uEdge* edgeListHead;
   uEdge* MSTlistHead;
   int totalMSTCost;

   PrimMST(int nodes, int setNode) {
      numNodes = nodes;
      nodeInSetA = setNode;
      whichSet = new int[nodes + 1];
      for (int i = 1; i <= numNodes; i++) {
         whichSet[i] = 2;
      }
      whichSet[0] = 1;
      whichSet[nodeInSetA] = 1;
```

```
        edgeListHead = new uEdge(0,0,0);
        edgeListHead->next = NULL;
        MSTlistHead = new uEdge(0,0,0);
        MSTlistHead->next = NULL;
        totalMSTCost = 0;
    }

    void listInsert(uEdge* newEdge){
        uEdge* edge = edgeListHead;

        while(edge->next != NULL && newEdge->cost > edge->next->cost) {
            edge = edge->next;
        }
        newEdge->next = edge->next;
        edge->next = newEdge;
    }

    uEdge* removeEdge() {
        uEdge* e = edgeListHead;
        uEdge* temp = edgeListHead;
        while (e != NULL) {
            if ((whichSet[e->next->Ni] != whichSet[e->next->Nj]) && (whichSet[e->next->Ni] == 1
|| whichSet[e->next->Nj] == 1)) {
                temp = e->next;
                e->next = e->next->next;
                return temp;
            }
            e = e->next;
        }
        return temp;
    }

    void addEdge(uEdge* edge) {
        edge->next = MSTlistHead->next;
        MSTlistHead->next = edge;
    }
    void printSet(ofstream& debugFile) {
        debugFile << "whichSet: ";
        for (int i = 0; i <= numNodes; i++) {
            debugFile << whichSet[i] << " ";
        }
        debugFile << endl;
    }
    void printEdgeList(ofstream& debugFile) {
        uEdge* temp = edgeListHead;
        int count = numNodes;
        debugFile << "EdgeListHead --> ";
        while(temp != NULL && count > 0) {
            temp->printEdge(temp,debugFile);
            debugFile << " -> ";
            temp = temp->next;
            count--;
        }
        debugFile << "NULL" << endl << endl;
    }
    void printMSTList(ofstream& debugFile) {
        uEdge* temp = MSTlistHead;
        int count = numNodes;
        debugFile << "MSTListHead --> ";
        while(temp != NULL && count > 0) {
            temp->printEdge(temp,debugFile);
            debugFile << " -> ";
            temp = temp->next;
            count--;
        }
        debugFile << "NULL" << endl << endl;
    }
    void updateMST(uEdge* newEdge) {
        addEdge(newEdge);
        totalMSTCost += newEdge->cost;

        if (whichSet[newEdge->Ni] == 1)
            whichSet[newEdge->Nj] = 1;
        else
            whichSet[newEdge->Ni] = 1;
    }
    bool setBisEmpty(){
        for(int i = 1; i <= numNodes; i++) {
            if(whichSet[i] == 2) return false;
        }
```

```
        return true;
    }
};

int main(int argc, const char * argv[]) {

    string inputName = argv[1];
    ifstream input;
    input.open(inputName);

    string nodeInSetA = argv[2];

    string outputName1 = argv[3];
    ofstream MSTfile;
    MSTfile.open(outputName1);

    string outputName2 = argv[4];
    ofstream debugFile;
    debugFile.open(outputName2);

    int nodes;
    input >> nodes;

    PrimMST prim(nodes, stoi(nodeInSetA));
    prim.printSet(debugFile);
    int ni;
    int nj;
    int cost;
    while (!input.eof()) {
        input >> ni >> nj >> cost;
        uEdge* newEdge = new uEdge(ni,nj,cost);
        prim.listInsert(newEdge);
        prim.printEdgeList(debugFile);
    }
    while(!prim.setBisEmpty()) {
        uEdge* newEdge = prim.removeEdge();
        debugFile << "Preforming Remove Edge" << endl;
        newEdge->printEdge(newEdge, debugFile);
        debugFile << endl;
        prim.updateMST(newEdge);
        prim.printSet(debugFile);
        prim.printEdgeList(debugFile);
        prim.printMSTList(debugFile);
    }

    MSTfile << "*** Prim's MST of the input graph G is: ***" << endl;
    MSTfile << prim.numNodes << endl;
    prim.printMSTList(MSTfile);
    MSTfile << "*** MST total cost = " << prim.totalMSTCost << " ***" << endl;


    input.close();
    MSTfile.close();
    debugFile.close();

    return 0;
}
```

Set A = 1 MST File

```
*** Prim's MST of the input graph G is: ***
12
MSTListHead --> <0, 0, 0, 3> -> <3, 2, 25, 9> -> <9, 11, 15, 9> -> <9, 8, 21, 8> -> <8, 10, 15,
3> -> <3, 5, 14, 8> -> <8, 6, 12, 5> -> <5, 4, 12, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 6> -> <6, 12, 8,
1> -> <1, 6, 23, NULL> -> NULL

*** MST total cost = 162 ***
```

Set A = 1 Debug File

```
whichSet: 1 1 2 2 2 2 2 2 2 2 2 2 2
EdgeListHead --> <0, 0, 0, 6> -> <6, 4, 13, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 6> -> <6, 4, 13, 12> -> <12, 7, 24, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 6> -> <6, 12, 8, 6> -> <6, 4, 13, 12> -> <12, 7, 24, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 6> -> <6, 12, 8, 6> -> <6, 4, 13, 10> -> <10, 12, 17, 12> -> <12, 7,
24, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 6> -> <6, 12, 8, 6> -> <6, 4, 13, 10> -> <10, 12, 17, 12> -> <12, 7,
24, 9> -> <9, 10, 41, NULL> -> NULL
```

EdgeListHead --> <0, 0, 0, 6> -> <6, 12, 8, 6> -> <6, 4, 13, 10> -> <10, 12, 17, 12> -> <12, 7, 24, 2> -> <2, 4, 31, 9> -> <9, 10, 41, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 6> -> <6, 12, 8, 6> -> <6, 4, 13, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 12> -> <12, 7, 24, 2> -> <2, 4, 31, 9> -> <9, 10, 41, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 6> -> <6, 12, 8, 6> -> <6, 4, 13, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> <2, 4, 31, 9> -> <9, 10, 41, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 6> -> <6, 4, 13, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> <2, 4, 31, 9> -> <9, 10, 41, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 6> -> <6, 4, 13, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 1> -> <1, 6, 23, 12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> <2, 4, 31, 9> -> <9, 10, 41, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 8> -> <8, 6, 12, 6> -> <6, 4, 13, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 1> -> <1, 6, 23, 12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> <2, 4, 31, 9> -> <9, 10, 41, NULL> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 8> -> <8, 6, 12, 6> -> <6, 4, 13, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21, 1> -> <1, 6, 23, 12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> <2, 4, 31, 9> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 8> -> <8, 6, 12, 6> -> <6, 4, 13, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21, 1> -> <1, 6, 23, 12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 5> -> <5, 4, 12, 8> -> <8, 6, 12, 6> -> <6, 4, 13, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21, 1> -> <1, 6, 23, 12> -> <12, 7, 24, 3> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 5> -> <5, 4, 12, 8> -> <8, 6, 12, 6> -> <6, 4, 13, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21, 4> -> <4, 3, 23, 1> -> <1, 6, 23, 12> -> NULL

```
EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 5> -> <5, 4, 12, 8> -> <8, 6, 12, 6> ->
<6, 4, 13, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21, 4> -> <4, 3, 23,
1> -> <1, 6, 23, 12> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 5> -> <5, 4, 12, 8> -> <8, 6, 12, 6> ->
<6, 4, 13, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21, 4> -> <4, 3, 23,
1> -> <1, 6, 23, 12> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 5> -> <5, 4, 12, 8> -> <8, 6, 12, 6> ->
<6, 4, 13, 3> -> <3, 5, 14, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21,
4> -> <4, 3, 23, 1> -> NULL

EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 6> -> <6, 7, 12, 5> -> <5, 4, 12, 8> ->
<8, 6, 12, 6> -> <6, 4, 13, 3> -> <3, 5, 14, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17,
9> -> <9, 8, 21, 4> -> NULL

Preforming Remove Edge
<1, 6, 23, 12>
whichSet: 1 1 2 2 2 2 1 2 2 2 2 2 2
EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 12, 8, 6> -> <6, 7, 12, 5> -> <5, 4, 12, 8> ->
<8, 6, 12, 6> -> <6, 4, 13, 3> -> <3, 5, 14, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17,
9> -> <9, 8, 21, 4> -> NULL

MSTListHead --> <0, 0, 0, 1> -> <1, 6, 23, NULL> -> NULL

Preforming Remove Edge
<6, 12, 8, 6>
whichSet: 1 1 2 2 2 2 1 2 2 2 2 2 1
EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 5> -> <5, 4, 12, 8> -> <8, 6, 12, 6> ->
<6, 4, 13, 3> -> <3, 5, 14, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21,
4> -> <4, 3, 23, 12> -> NULL

MSTListHead --> <0, 0, 0, 6> -> <6, 12, 8, 1> -> <1, 6, 23, NULL> -> NULL

Preforming Remove Edge
<6, 7, 12, 5>
whichSet: 1 1 2 2 2 2 1 1 2 2 2 2 1
EdgeListHead --> <0, 0, 0, 5> -> <5, 7, 5, 5> -> <5, 4, 12, 8> -> <8, 6, 12, 6> -> <6, 4, 13, 3> ->
<3, 5, 14, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21, 4> -> <4, 3, 23,
12> -> <12, 7, 24, 3> -> NULL
```

```
MSTListHead --> <0, 0, 0, 6> -> <6, 7, 12, 6> -> <6, 12, 8, 1> -> <1, 6, 23, NULL> -> NULL

Preforming Remove Edge
<5, 7, 5, 5>
whichSet: 1 1 2 2 2 1 1 1 2 2 2 2 1
EdgeListHead --> <0, 0, 0, 5> -> <5, 4, 12, 8> -> <8, 6, 12, 6> -> <6, 4, 13, 3> -> <3, 5, 14, 8>
-> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21, 4> -> <4, 3, 23, 12> -> <12,
7, 24, 3> -> <3, 2, 25, 2> -> NULL

MSTListHead --> <0, 0, 0, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 6> -> <6, 12, 8, 1> -> <1, 6, 23,
NULL> -> NULL

Preforming Remove Edge
<5, 4, 12, 8>
whichSet: 1 1 2 2 1 1 1 1 2 2 2 2 1
EdgeListHead --> <0, 0, 0, 8> -> <8, 6, 12, 6> -> <6, 4, 13, 3> -> <3, 5, 14, 8> -> <8, 10, 15, 9>
-> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21, 4> -> <4, 3, 23, 12> -> <12, 7, 24, 3> -> <3,
2, 25, 2> -> <2, 4, 31, 1> -> NULL

MSTListHead --> <0, 0, 0, 5> -> <5, 4, 12, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 6> -> <6, 12, 8, 1> ->
<1, 6, 23, NULL> -> NULL

Preforming Remove Edge
<8, 6, 12, 6>
whichSet: 1 1 2 2 1 1 1 1 1 2 2 2 1
EdgeListHead --> <0, 0, 0, 6> -> <6, 4, 13, 3> -> <3, 5, 14, 8> -> <8, 10, 15, 9> -> <9, 11, 15,
10> -> <10, 12, 17, 9> -> <9, 8, 21, 4> -> <4, 3, 23, 12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> <2,
4, 31, 1> -> <1, 11, 36, 9> -> NULL

MSTListHead --> <0, 0, 0, 8> -> <8, 6, 12, 5> -> <5, 4, 12, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 6> ->
<6, 12, 8, 1> -> <1, 6, 23, NULL> -> NULL

Preforming Remove Edge
<3, 5, 14, 8>
whichSet: 1 1 2 1 1 1 1 1 1 2 2 2 1
EdgeListHead --> <0, 0, 0, 6> -> <6, 4, 13, 8> -> <8, 10, 15, 9> -> <9, 11, 15, 10> -> <10, 12,
17, 9> -> <9, 8, 21, 4> -> <4, 3, 23, 12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> <2, 4, 31, 1> -> <1,
11, 36, 9> -> <9, 10, 41, 1> -> NULL
```

```
MSTListHead --> <0, 0, 0, 3> -> <3, 5, 14, 8> -> <8, 6, 12, 5> -> <5, 4, 12, 5> -> <5, 7, 5, 6> ->
<6, 7, 12, 6> -> <6, 12, 8, 1> -> <1, 6, 23, NULL> -> NULL

Preforming Remove Edge
<8, 10, 15, 9>
whichSet: 1 1 2 1 1 1 1 1 1 2 1 2 1
EdgeListHead --> <0, 0, 0, 6> -> <6, 4, 13, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 9> -> <9, 8, 21,
4> -> <4, 3, 23, 12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> <2, 4, 31, 1> -> <1, 11, 36, 9> -> <9,
10, 41, 1> -> <1, 2, 46, NULL> -> NULL

MSTListHead --> <0, 0, 0, 8> -> <8, 10, 15, 3> -> <3, 5, 14, 8> -> <8, 6, 12, 5> -> <5, 4, 12, 5>
-> <5, 7, 5, 6> -> <6, 7, 12, 6> -> <6, 12, 8, 1> -> <1, 6, 23, NULL> -> NULL

Preforming Remove Edge
<9, 8, 21, 4>
whichSet: 1 1 2 1 1 1 1 1 1 1 1 2 1
EdgeListHead --> <0, 0, 0, 6> -> <6, 4, 13, 9> -> <9, 11, 15, 10> -> <10, 12, 17, 4> -> <4, 3, 23,
12> -> <12, 7, 24, 3> -> <3, 2, 25, 2> -> <2, 4, 31, 1> -> <1, 11, 36, 9> -> <9, 10, 41, 1> -> <1,
2, 46, NULL> -> NULL

MSTListHead --> <0, 0, 0, 9> -> <9, 8, 21, 8> -> <8, 10, 15, 3> -> <3, 5, 14, 8> -> <8, 6, 12, 5>
-> <5, 4, 12, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 6> -> <6, 12, 8, 1> -> <1, 6, 23, NULL> -> NULL

Preforming Remove Edge
<9, 11, 15, 10>
whichSet: 1 1 2 1 1 1 1 1 1 1 1 1 1
EdgeListHead --> <0, 0, 0, 6> -> <6, 4, 13, 10> -> <10, 12, 17, 4> -> <4, 3, 23, 12> -> <12, 7,
24, 3> -> <3, 2, 25, 2> -> <2, 4, 31, 1> -> <1, 11, 36, 9> -> <9, 10, 41, 1> -> <1, 2, 46, NULL>
-> NULL

MSTListHead --> <0, 0, 0, 9> -> <9, 11, 15, 9> -> <9, 8, 21, 8> -> <8, 10, 15, 3> -> <3, 5, 14,
8> -> <8, 6, 12, 5> -> <5, 4, 12, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 6> -> <6, 12, 8, 1> -> <1, 6, 23,
NULL> -> NULL

Performing Remove Edge
<3, 2, 25, 2>
whichSet: 1 1 1 1 1 1 1 1 1 1 1 1 1
EdgeListHead --> <0, 0, 0, 6> -> <6, 4, 13, 10> -> <10, 12, 17, 4> -> <4, 3, 23, 12> -> <12, 7,
24, 2> -> <2, 4, 31, 1> -> <1, 11, 36, 9> -> <9, 10, 41, 1> -> <1, 2, 46, NULL> -> NULL
```

```
MSTListHead --> <0, 0, 0, 3> -> <3, 2, 25, 9> -> <9, 11, 15, 9> -> <9, 8, 21, 8> -> <8, 10, 15,
3> -> <3, 5, 14, 8> -> <8, 6, 12, 5> -> <5, 4, 12, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 6> -> <6, 12, 8,
1> -> <1, 6, 23, NULL> -> NULL
```

Set A = 4 MST File:

```
*** Prim's MST of the input graph G is: ***
12
MSTListHead --> <0, 0, 0, 3> -> <3, 2, 25, 1> -> <1, 6, 23, 9> -> <9, 11, 15, 9> -> <9, 8, 21, 8>
-> <8, 10, 15, 3> -> <3, 5, 14, 8> -> <8, 6, 12, 6> -> <6, 12, 8, 6> -> <6, 7, 12, 5> -> <5, 7, 5, 5>
-> <5, 4, 12, NULL> -> NULL

*** MST total cost = 162 ***
```

Set A = 8 MST File:

```
*** Prim's MST of the input graph G is: ***
12
MSTListHead --> <0, 0, 0, 3> -> <3, 2, 25, 1> -> <1, 6, 23, 9> -> <9, 11, 15, 9> -> <9, 8, 21, 8>
-> <8, 10, 15, 3> -> <3, 5, 14, 5> -> <5, 4, 12, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 6> -> <6, 12, 8, 8>
-> <8, 6, 12, NULL> -> NULL

*** MST total cost = 162 ***
```

Set A = 12 MST File:

```
*** Prim's MST of the input graph G is: ***
12
MSTListHead --> <0, 0, 0, 3> -> <3, 2, 25, 1> -> <1, 6, 23, 9> -> <9, 11, 15, 9> -> <9, 8, 21, 8>
-> <8, 10, 15, 3> -> <3, 5, 14, 8> -> <8, 6, 12, 5> -> <5, 4, 12, 5> -> <5, 7, 5, 6> -> <6, 7, 12, 6>
-> <6, 12, 8, NULL> -> NULL

*** MST total cost = 162 ***
```