```
/* CMPUT 201 Assignments                                                    */
/* Dues:                                                                    */
/* #2:    12:30pm, March 8, 2016;                                           */
/* #3:    12:30pm, April 7, 2016                                            */
```

(Mandatory assignment cover-sheet; without it, your work will not be marked.)

Submitting student: Justin Barclay

Collaborating classmates: _____

Other collaborators: _____

References: StackOverflow, The C Programming Book, Modern C Programming

---

Regardless of the collaboration method allowed, you must always properly acknowledge the sources you used and people you worked with. Your professor reserves the right to give you an exam (oral, written, or both) to determine the degree that you participated in the making of the deliverable, and how well you understand what was submitted. For example, you may be asked to explain any solution that was submitted and why you choose to write it that way. This may impact the mark that you receive for the deliverable.

So, whenever you submit a deliverable, especially if you collaborate, you should be prepared for an individual inspection/walkthrough in which you explain what every line of assignment does and why you choose to write it that way.

```
/* CMPUT 201 Assignments                                                    */
/* Dues:                                                                    */
/* #2:    12:30pm, March 8, 2016;                                           */
```

Given a set of n points on the 2-dimensional plane, the *rectilinear Steiner tree* (RST) problem is to find the rectilinear tree in the plane, which connects the given set of points and has the minimum total wire length (referred to as the cost of the tree). The RST problem has many applications in VLSI physical circuit design, however it is proven to be NP-hard[1]. The project is on a heuristic algorithm that first computes a *minimum-weight spanning tree* (MST)[2] on the given set of points, then carefully lays out the edges of the MST such that the total length of the overlapping portions of the wires is maximized. Essentially an overlapping portion of two (or more) wires can be merged into one wire while introducing extra non-given points (called *Steiner* points), and thus to obtain an approximate rectilinear Steiner tree of a hopefully low cost.

Our goal is to design a C program that performs all these tasks, and to empirically test the quality of the achieved approximate rectilinear Steiner tree in terms of the cost reduction from the MST. This project is thus a typical and important step in most scientific research, called "numerical experiments".

The project is decomposed into three parts, each becomes an assignment worth 10%.

Detailed specifications for Assignment #2:

1. Recall that your C program uses option "`-i`" to accept an input file whose name directly follows the option "`-i`"; the input file specifies an instance I, which contains n given points $p_i = (x_i, y_i)$, $i = 1, 2, \ldots, n$.

2. For any two points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$, the rectilinear (also called Manhattan, or L_1) distance between them is $d(p_i, p_j) = |x_i - x_j| + |y_i - y_j|$, which is also called the weight of the edge (p_i, p_j).

3. The major task in this assignment is to compute the minimum-weight spanning tree (MST) on these n given points, by Prim's algorithm. Prim's algorithm can be described as follows:

   - The algorithm basically "grows" the MST by including one point in an iteration.
   - The algorithm starts with an arbitrary point, say p_{i_0}.
   - At the beginning of each iteration, let T denote the current partial MST, let V denote the subset of points in T, and S denote the subset of points outside T.
   - For each point of S, say p_i, its distance to T is calculated as the minimum distance from p_i to all the points of V, *i.e.* $d_i = \min_{p_j \in V} d(p_i, p_j)$; then p_{i^*} is selected to include in this iteration if its distance to T is minimized over all points of S, *i.e.* $d_{i^*} = \min_{p_i \in S} d_i$; and the iteration ends.
   - In the classic version of Prim's algorithm, there could be ties of multiple points of S having the same distance to T and could be ties of multiple edges of the same weight from a specific point to T; the ties are broken arbitrarily.

     For our purpose, we **break the ties in the following way** (even if $i^* = j^*$):

     1) assume $d_{i^*} = d(p_{i^*}, p_i)$ and $d_{j^*} = d(p_{j^*}, p_j)$ are both minimum;
        edge (p_{i^*}, p_i) is selected to include point p_{i^*} (move p_{i^*} from S to V),

[1] Want to know what it means? Come to my CMPUT 304 class!

[2] You likely encountered it in the past, or you will see it in CMPUT 204.

2) if $|y_{i^*} - y_i| > |y_{j^*} - y_j|$,
3) or, if $|y_{i^*} - y_i| = |y_{j^*} - y_j|$ and $\max\{x_{i^*}, x_i\} > \max\{x_{j^*}, x_j\}$,
(otherwise select either edge to include the corresponding point).

4. There are two key properties of the MST computed by this version of Prim's algorithm (no need from you to prove them, but validate them if possible):

   - the maximum degree of the MST is at most 8;
   - the tree is *separable*, meaning that any two non-adjacent edges of the tree do not overlap, disregarding how you lay them down on the plane.

   These properties enable us to optimally lay out the edges of the computed MST to maximize the total length of the overlapping portions (Assignment #3).

5. For Assignment #2, append the edges of the MST to the input instance, one at a line:

   - assume the n points are indexed in sequential order, from 1 to n;
   - add a comment line:
     `# edges of the MST by Prim's algorithm:`
   - an edge is printed as:
     `i j weight`
     where `i` and `j` are the indices of the two end-points of the edge, and `weight` is the distance between them. These three values are separated by a white space.

6. Results that you do not need to append to the input instance file but you need to collect include the following (useful for Assignment #3):

   - the total weight of the MST (*i.e.*, the linear sum of the edge weights in the tree);
   - the degree of each point.

//End of Assignment #2.