```
/* CMPUT 201 Assignments                                                    */
/* Dues:                                                                    */
/* #1:    12:30pm, February 2, 2016;                                        */
/* #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 (ID# 1237448)

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:                                                                    */
/* #1:    12:30pm, February 2, 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 #1:

1. Let us fix the plane area to be $[0, \texttt{MAX\_X}] \times [0, \texttt{MAX\_Y}]$, representing the circuit board, where `MAX_X` and `MAX_Y` are integers. In the entire project we assume all given points have the integer coordinates (but your C program should leave room to handle other data types).

2. An input file describes an instance of the RST problem.

   The filename convention is "`instanceXXX_YYY.txt`", where `XXX` is the number of given points (denoted as `NUM_PT` in the sequel) and `YYY` is the index. Inside the file, every line starting with a symbol "`#`" is a comment. Other than comment lines and blank lines (which are only for separation purpose), the first line contains two numbers which are `MAX_X` and `MAX_Y`, the second line contains one number `NUM_PT`, and each of the rest (exactly) `NUM_PT` lines specifies the x-coordinate and y-coordinate of a given point. Numbers in the same line are separated by white spaces.

   Your C program uses option "`-i`" to accept an input file whose name directly follows the option "`-i`"; if input file option is not used, your C program works to generate a number of (random) instances. To this purpose, it reads input from stdin with the following appearance:

   ```
   Generating random instances ...
   Enter the circuit board size MAX_X MAX_Y:
   Enter the number of points NUM_PT:
   Enter the number of random instances to be generated:
   ```

   To form an instance, a total of `NUM_PT` non-duplicating points are randomly generated by your C program, and the instance is written into a file with the filename convention above. This repeats for the number of times read from stdin, and `YYY` is incremented from `1` to this number.

   //End of Assignment #1.

[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.