EECS65– Computational Geometry
Lab assignment # 2 – Fall 2016

Important Information

1. This lab is due October 10, 2016 (strictly observe the deadline in CatCourses).
2. Your solution must be electronically submitted to CatCourses Crops.
3. This programming assignment **must** be solved individually.
4. You can code your solution in Matlab, C/C++, or Java. The choice is up to you. But you have to properly document your submission to explain how your program can be compiled and run. **Important**: if you solve your assignment in C/C++, it **must** compile with `gcc`. Other languages will not be accepted. Do not include solutions depending on the availability of IDEs (e.g., Eclipse or similar). If you use C/C++ or java, please provide instructions to compile your code from the command line.
5. Submissions that do not compile will receive 0 points without further consideration.
6. Your code will be closely inspected to see whether you followed the algorithm presented in class. Solutions not following the algorithm presented in class will receive 0 points.

Segment intersection problem

Implement the segment intersection algorithm studied in class (Chapter 2 in the textbook). The input to the algorithm is provided in a text file called `input.txt` with n lines, where n is the number of segments. Each line consists of four numbers giving the x and y coordinates of the vertices of the segment. For example

```
1 3.5 5 9.45
```

describes a segment between points $p_1 = (1, 3.5)$ and $p_2 = (5, 9.45)$ Note that numbers in a line are separated by a single space.

Your program should write its results to an output file called `output.txt`. Each line of the output should start with the coordinates of a detected intersection point, and then the indexes of the segments involved in the intersection follow. For example

```
3.4 5.6 4 7 22
```

means that segments s_4, s_7 and s_{22} intersect at point $p = (3.4, 5.6)$. Note that segment s_4 is the segment appearing in the fourth line of `input.txt`, s_7 appears in the seventh line, and so on.

- Your code should correctly deal with degenerate cases, like vertical or horizontal segments. Refer to the textbook for details.
- Depending on the language you choose to code your solution, binary search trees may be readily available in a standard library or not. You are welcome and encouraged to use libraries when available. In that case, you can refer to the documentation to discuss the efficiency of the implementation you borrowed.