CSE 2320 Lab Assignment 1

Due September 22, 2016

Goals:

1. Understanding of mergesort.

2. Understanding of binary search.

Requirements:

1. Design, code, and test a C program to 1) mergesort a set of 2-D points based on their angles around the origin and 2) uses binary search to determine the number of points at the same angle (e.g. ray) as a given point. Your program may reuse available code (e.g. from Notes 1). The first line of the input will be n, the number of target 2-D points on the next n input lines. The line after the last input point will contain m, the number of 2-D query points on the next m input lines. Each coordinate should be read as an `int`. The input should be read from standard input using `scanf` (see `LIS.c` on the course webpage). The output is one line per query point that echoes the coordinates of the query point and the number of target points on the ray of the query point.

2. Submit your program on Blackboard by 1:45 pm on September 22. One of the comment lines should indicate the compilation command used on OMEGA.

Getting Started:

1. The input will not include the orgin as a point.

2. One way to implement the sort is to convert cartesian (x, y) coordinates to polar coordinates (r, θ) where r is positive and $0 \le \theta < 2\pi$. Since this is inherently ***slow*** (e.g. computing the arctangent with `atan2()`), and possibly ***inaccurate***, you ***must*** do all computations on integers ***without*** using division. It is still useful, however, to think in terms of slopes and apply a little algebra. It is also useful to coarsely classify points based on the signs of x and y.

3. If a duplicate target point is discovered while merging, keep the one that was earlier in the input sequence. (So n may be decreased.) Print a message indicating the element that has been discarded.

   If two points on the same ray are discovered while sorting, the one ***closer*** to the origin will appear earlier in the output array than the other point.

4. Your program ***should not*** prompt for an input file name. Instead, a ***shell redirect*** (`a.out < file1.dat`), typing the input, or cut-and-paste should be used when testing your program. The mechanism for input redirection makes the referenced file appear to the executing program as though its contents are being typed manually at the keyboard.

5. All input coordinates will be in the range `-32767 .. 32767`

6. Arrays should be allocated dynamically.

7. Each query is to be processed in $\Theta(\log n)$ time.