CSE 2320 Lab Assignment 2

Due July 13, 2011

Goal:

Understanding of dynamic programming, especially longest increasing subsequences.

Requirements:

You have received a government contract to enhance the Personal Artillery Device (PAD). Since the PAD has been shown to be too cumbersome for its original intent, destroying aerial targets, its deployment is being limited to destroying ground targets in a flat field. In its new configuration, the PAD will interface with a PRU (Personal RADAR Unit) and a PCU (Personal Computation Unit) to reduce the number of operator decisions. Each mission consists of:

a. Moving an integrated PAD/PRU/PCU to a position (the origin) within a field.
b. Position the gun facing due East at a nearly vertical position.
c. Load the projectiles.
d. Applying the following operations to destroy targets (as identified by the PRU):
    1. Rotate the gun counterclockwise (previous experience suggests that gears will wear rapidly if clockwise motion is also allowed).
    2. Lower the gun towards horizontal (previous experience suggests that it is difficult to raise a loaded gun).
    3. Fire a projectile.
e. Before the PAD has rotated 360°, the operator must pick up everything and run to avoid the return fire.

This specification suggests that the sequence of selected targets will begin with targets at *increasing* distance from the PAD and then, after lowering the gun below 45°, finish with targets at *decreasing* distance from the PAD (see `http://en.wikibooks.org/wiki/High_school_physics/Projectile_motion` for the precise physics). Your mission is to compute a longest such sequence.

1. Design, code, and test a Java program to determine the points in a sequence satisfying the specification. Your program may reuse code from Lab 1 along with code for finding a longest increasing subsequence. The first line of the input will be *n*, the number of 2-D points on the next *n* input lines. Each coordinate should be read as an `int`. The input should be read from standard input using `java.util.Scanner` (especially `nextInt()`, see `LIS.java` on the course webpage). *Note that the input points are already rotationally ordered and there is never more than one point on a ray. All points are within the range of the* PAD. The first output line gives the number of points in the selected target sequence and the remaining lines give the target points in counterclockwise order, one per line.

2. Email your program to `hafizfahad@mavs.uta.edu` by 12:45 pm on July 13.

Getting Started:

1. The origin is never an input point.

2. All computations are to be done using integer arithmetic. Square roots are unnecessary.

3. Your program ***should not*** prompt for an input file name.