CSCI 135 Software Design and Analysis, C++
Homework 6
Due 3/28/2014

Saad Mneimneh
Hunter College of CUNY

Problem 1: A Point class
Consider the following class declaration for a point in 2D:

```
class Point {
  double x_coord;
  double y_coord;

 public:
  Point(double x, double y);
  void moveTo(double x, double y);
  double x();
  double y();
  void print(); //outputs x_ followed by ','
};              //followed y_ followed by new line
```

(a) Implement the five public functions including the constructor.

(b) Modify the class so that the following will compile:

```
int main() {
  const int n=...; //some value
  Point a[n];
  .
  .
  .
}
```

(c) Add a public function for class Point called seesToLeft with the following signature:

```
bool seesToLeft(Point p, Point q)
```

Imagine yourself standing at a point a. A function call a.seeToLeft(b,c) should return true iff when walking towards point b, point c is on your left. We will not worry about three points being on the same line. It is not obvious how to implement this function, so here's the condition for this to be true:

```
(a.y()-c.y())*(a.x()-b.x())>(a.y()-b.y())*(a.x()-c.x())
```

(d) Declare an array of 50 points randomly situated in the square with corners (0,0) and (100,100). To generate a random number in $[0, U]$ use

```
(double)rand()/RAND_MAX*U
```

Problem 2: Zoolander

This problem is inspired by a movie with the same title about a fashion model who cannot turn left. Given an array of points, the goal is to sort the array in a special way: if we have a walk that goes through $a[0], a[1], \ldots, a[n-1]$, we never make a left turn. We can always start at the lowest point, i.e. the one with the smallest y coordinate. Let's call this the zoolander sorting.

(a) Write a function called lowest that takes as parameters an array of points and its length and returns the index of the lowest point in the array.

(b) Write a function called swap that takes as parameters an array a of points, and two indices i and j, and swaps the two points $a[i]$ and $a[j]$.

(c) Write a function called next that takes as parameters an array a of points, and two indices i and j and returns an index k in $[i, j]$ such that no l in $[i, j]$ makes $a[i-1]$.seesToLeft$(a[k], a[l])$ true.

(d) Using parts (a), (b), and (c), zoolander sort the array that you create in Problem 1. Then display all the points in the array, cut and paste in excel, and verify using a scatter line chart that the walk makes no left turns, i.e. it is a clockwise spiral to the center.

Problem 3: A rectangle learning game
Consider the following class:

```
class Rectangle {
  Point LL;  //lower left corner of rectangle
  Point UR;  //upper right corner of rectangle

 public:
  Rectangle() {...}              //creates a random rectangle
                                 //inside the square with
                                 //corners (0,0) and (100,100)

  bool contains(Point p) {...}   //returns true iff p inside rectangle
};
```

(a) Complete the implementation of the class Rectangle.

(b) Observe that once you create a rectangle, there is no way to tell what that rectangle is. Write a program that first creates a rectangle and then attempts to learn what the rectangle is by using the member function contains repeatedly on random sample points. The idea is to keep track of the largest rectangle that is contained in the original one.

(c) Define the probability of error to be the probability that a point in the original rectangle lies outside your learned rectangle. Let R1 be the original rectangle and R2 your learned rectangle. Therefore,

$$Prob(\text{error}) = \frac{\text{number of points in R1 but not in R2}}{\text{number of points in R1}}$$

Compute this probability. Experiment with different numbers of sample points and observe their effect on the probability of error.