Homework 4, CSCE 240, Fall 2016

Overview

This is an exercise in recursion.

A standard dominoes game has 91 tiles, one for all the combinations of (a, b) for $0 \leq a, b, \leq 12$.

One standard version of a game of dominoes is as follows. Each player is "dealt" twelve dominoes. Starting with the "double n" domino, players place dominoes on end so that the adjacent halves have the same number of dots.

For example, one might have a chain of dominoes:

$$(6, 6)(6, 4)(4, 2)(2, 10)$$

which would then continue as long as a player had dominoes in her hand that could be played.

This Assignment

You initialize the game by creating a `vector` of `Domino` class instances that has all the dominoes in it. You will then "deal" a second vector with 12 randomly chosen dominoes (no repeats).

For each of the 12 possible starting values, find the longest chain of dominoes that can be made using the dominos in your hand.

For each of the 12 possible starting values, you should output the "new max" sequence every time you get a sequence of equal or greater length.

Part 4A

Your assignment for the 4A part of this is to generate a random deal using the random number functions and the appropriate data structures in the code.

The goal in the A part of the assignments is to ensure that when you start to do the real work of the computation you have valid data with which to work. Thus, being able to deal (and dump to the output) one legitimate deal of dominos is the first step in the programming process.

Some Issues

This requires the use of pseudo-random numbers. Pseudo-random numbers are generated using a deterministic (and thus not "random") process. Generally, the use of "random" numbers in a computer program is this. A deterministic function is called that will produce the same sequence of numbers each time (assuming the same seed value) so that you can test your code. If you got a different set of numbers each time, you would have to look closely at the data every time to see if your code was working. With the same sequence of numbers, you at least know what you should be seeing in the output.

However, the pseudo-random (actually "pseudo-random") numbers generated by this deterministic process are guaranteed (by the provider of the software) to satisfy the same statistical tests that random numbers would satisfy, and thus they can be used as "random" numbers.

Bottom line: You will get different numbers on a Mac, on the Linux machines, and on a Windows machine.

This is a good example of learning to be flexible when testing. I have provided you with a sequence of domino subscripts, produced with a certain set of seeds, and generated on a Mac.

If you comment out the part of your code that invokes the random number generator, and simply input these subscripts, then you will get test data that is the same as the data for which I have provided output. This will allow you to verify that your program does the same thing that mine does.

You must remember to un-comment this part, and to generate actual random numbers in your code, because when I run these programs for a grade I will supply a seed, generate a different sequence of subscripts on the Linux machines, and test using that.