University of Washington Bothell

CSS 342: Data Structures, Algorithms, and Discrete Mathematics

Program 2: Recursion

Purpose

The programming assignment will provide an exercise in using recursion. There are two problems which need to be solved. The first asks for a recursive function to calculate a Catalan number. This problem will also require the student to utilize command line arguments for their program. The second is to solve a path-finding problem in two dimensional space. The second problem will require the usage of recurrence as well as good class design.

Problem 1: The good Mr. Catalan.

The Catalan number form a sequence of natural numbers that occur in many counting problems. They are named after the mathematician Eugene Charles Catalan (http://en.wikipedia.org/wiki/Eug%C3%A8ne_Charles_Catalan) and are defined by the following recursive formula:

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \sum_{i=0}^{n} C_i \, C_{n-i} \quad \text{for } n \geq 0;$$

Write a program called Catalan which takes one argument and calls a recursive function which computes the nth Catalan number. The program then prints out the result to std::out.

For instance,

Catalan 4

Would print out: 14

Catalan 10

Would print out: 16796

Problem 2: "The Greedy Robot" or "Lost in the Supermarket"

A robot is positioned on an integral point in a two-dimensional coordinate grid (x_r, y_r). There is a treasure that has been placed at a point in the same grid at (x_t, y_t). All x's and y's are integers. The robot can move up (North), down (South), left (West), or right (East). Commands can be given to the robot to move one position in one of the four direction. That is, "E" moves a robot one slot East (to the right) so if the robot was on position (3, 4), it would now be on (4, 4). The command N would move the robot one position north so a robot at position (4, 4) would be at (4, 5).

Because the robot cannot move diagonally, the shortest distance between a robot at (x_r, y_r) and a treasure at (x_t, y_t) is

$$| x_r - x_t | + | y_r - y_t | = \text{ShortestPossibleDistance}$$

Write a recursive program which determines all the unique shortest possible paths from the robot to the treasure with the following stipulation: *The robot may never move in the same direction more than two times in a row.*

The input to the program will be the starting position of the robot (x_r, y_r) and the position of the treasure (x_t, y_t). These will be in read from cin as four integers. For instance, an input of 1 3 -2 4 corresponds to the robot starting at position (1, 3) and needing to get to position (-2, 4). Do not worry about error conditions on input as we will assume the input is well formed. Read the input from cin.

The output of the program should be the listing of all the unique shortest possible paths followed by the number of unique paths. The paths must follow the stipulation whereby the robot cannot move in the same direction more than two times in a row. A path should be output as a string of characters with each character corresponding to a direction the Robot should move. For instance, NNENE corresponds to having the robot move North, North, East, North and East. This would be one answer to the input: 3 3 5 6, which corresponds to (3,3) -> (5,6).

Notice that not all combinations of robots / treasures will have a solution. As there may not be a ShortestPossibleDistance given the stipulation. For instance, 3 3 3 7 has shortest possible distance of 4 but no way to get there in that distance without going North more than 2 times in a row.

For the sake of efficiency, do not make two separate recursive calls in your program (one to count and one to print). Make sure that one recursive call handles both.

For the input 1 2 3 5 which corresponds to (1,2) -> (3,5) the output should be:

NNENE
NNEEN
NENNE
NENEN
NEENN
ENNEN
ENENN
Number of paths: 7

Turn-In

- In a .ZIP file
  - All .cpp / .h files required
  - Either exe (if built on windows) or a.out if built on linux
  - One executable for each program
    - Catalin
    - GreedyRobot