Assignment DS4

Due Date: March 27

Purpose

The main purpose of this assignment is to help you slide into the wonderful world of recursion solving a real-world problem. The recursive solutions run the gamut from easy (input and output) to more challenging (finding a path; see below). You should also practice using good OOP techniques, meaning that you should design a class containing good private/public members and reuse code as much as possible.

Problem

The United States Geological Survey (USGS) is the mapping agency for the country. One of the most challenging problems is to guide an autonomous vehicle from one point to another, following the "best" path. This "best" path can be defined in various ways, such as the straightest path, the quickest, the one with the smallest slope, etc.

The problem with such a vehicle is that software is needed to guide it. You have been enlisted to write a search algorithm in C++ that will guide the vehicle through any terrain automatically. Finding a best path is inherently difficult; recursion is used routinely in such applications. You will write a recursive search algorithm that will map out a path from one point (source) to a destination in a given terrain map.

Input

Your program should prompt the user for the name of a PGM file. The program should then read the file which contains a header as follows:

`P2`
C R
`255`

where C is the number of columns and R is the number of rows. Note that it is possible that `C`$\neq$`R`. What follows is R rows of C integers representing gray-scale values. These also can be thought of as relative elevations. A sample input file is available on the course web page and is shown in Figure 1. The program should prompt the user for the PPM output file name and then then for the starting point row S, where $0 \leq S < R$ (no error checking required), and finally another output file name for the PPM that includes the search path; see *Specifics* section below.

Output

The program should turn the gray-scale PGM file into a color PPM file. This should be done by simply by the following formulas:

$R = \frac{GS}{2}$, $G = GS$, and $B = \frac{GS}{2}$

where R, G, B are the red, green, and blue values, respectively, of the PPM file and GS represents a gray-scale value of the PGM file. The format of the PPM file is the same as the PGM, except `P3` should be the first line of the header instead of `P2`.

The vehicle should search for the best path from the west side (column 0) at the given row to anywhere on the east side (last column). The output should be the PPM file rendered in green hues, followed by another PPM file that includes the search path the vehicle took from the starting point to any row in the last column. From any position, the vehicle can only move up, down, or to the right. The best path is defined as following the path that has the least elevation gain at each

step, where the relative elevation is assumed to be the same as the gray-scale value. The PPM file that is generated should be the same as before but with the path shown in red, as shown in Figure 2.

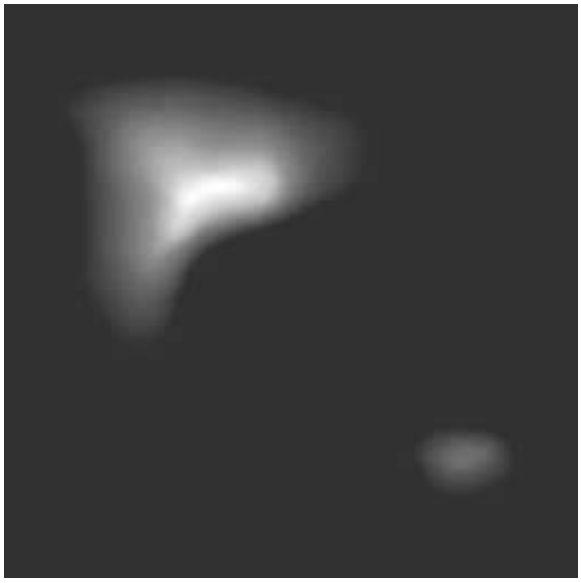

Figure 1: Input PGM file.

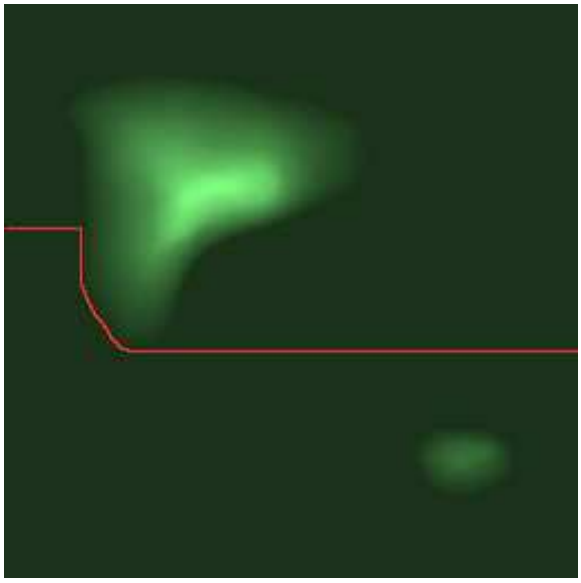

Figure 2: Generated PPM file with path starting at row 100.

Finally, the total elevation gain/loss should be displayed.

Specifics

- You must create a class called `terrain`. All of the methods and data must belong to this class.
- The constructor should read the file and initialize your data structure. You may wish to call an additional method from the constructor.
- Your class should contain the following public methods:

writePPM(*filename*)	–	write the PPM to given *filename*
searchPath()	–	find path through the terrain; update the PPM
writePathPPM(*filename*)	–	write the PPM with the path to given *filename*
elevationGainLoss()	–	return the total elevation gain (positive)/loss (negative)

  Any additional methods should be private.
- Here's the big one: **NO LOOPS OF ANY KIND are allowed in this program!** *All* of your repetition must be done with recursion.
- Your program should promote code reuse. That is, identical or very similar code should not be repeated from one function/method to another. Take advantage of OOP's capabilities and features.

- Your `main()` should be a test program; it should be very short and call the above methods exactly as follows:

```
int main () {
   string filename;

   cout << "Enter PGM filename: ";
   cin >> filename;
   terrain myTerrain(filename);

   cout << "Enter PPM filename: ";
   cin >> filename;
   myTerrain.writePPM(filename);
   cout << "Searching for path..." << endl;
   myTerrain.searchPath();
   cout << "Enter filename for PPM with path: ";
   cin >> filename;
   myTerrain.writePathPPM(filename);
   cout << "Total elevation gain/loss: ";
   cout << myTerrain.elevationGainLoss() << endl;
   return 0;
}
```

  Note that the only argument passed to any of the public methods is a filename.

Notes

As usual, write one method at a time. Begin by reading a small sample PGM and then displaying it to make sure the program reads the data correctly. Then add other methods one at a time.

The solution to this problem is not very long, due to the use of recursion. However, it may not be particularly easy, due to the use of recursion!

Submit your source code the usual way, using the usual naming conventions. Turn in hard copy of your code at the beginning of class on March 28^{th}.

At the recurrent end of the unending...
– T.S. Eliot