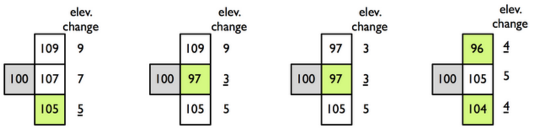
Due Sept 15, 11.59pm

This project is adapted from the Nifty projects (`http://nifty.stanford.edu/2016/franke-mountain-paths/`) and is an exercise navigating a 2D array, drawing images, and an example of a greedy algorithm.

You are given elevation data of a mountainous region in the form of a 2D array of integers (see example image below of one of the dataets). Your goal is to find a path that takes you through the points with the lowest elevation changes, in an effort to minimize the overall effort in walking through the path. For this you will use a 'greedy' approach and make local decisions in determining the successive points in the path.



Case 1: smallest change is 5. Go Down.

Case 2: smallest change is 3. Go forward.

Case 3: Tie. Smallest change is 3. Forward is possible so go forward.

Case 4: Tie. Smallest change is 5. Flip a coin to go up or down.

See the figure above. The idea is to start from an edge of the image (say the leftmost column), then make moves based on the pixels to the right, each time choosing the pixel that results in the smallest change. Your goal is to reach the right edge of the image.

Datasets: Datasets of multiple sizes will be provided. All are text files and will contain the width and height at the beginning of the file followed by the elevation values.

Tasks.

1. **Read dataset, create image.** Use file I/O (ifstream) to read the dataset. You will read the dataset into a 1D array (similar to project 1). You will use 1D array addressing in part 2 (below). After reading the dataset, find the maximum value in the image as it will be needed for writing out the image. You will then write out the data in the PPM format (see `http://paulbourke.net/dataformats/ppm/`) for more information about this format. The format is very similar to the the input file but with a more descriptive header and RGB triples for each pixel:

   P3
   width height
   MaxVal val1 val1 val1 val2 val2 val2 val2 val3 val3 .... val N val N val N

   where width and height are the dimensions of the image, MaxVal is the maximum value in the image and is followed by each pixel value (row major, the same as the input dataset order, carriage returns are optional, but a space between successive values is required). We write out the gray value as RGB triples (they are all the same value in this case). You should see an image like this (for the colorado480x480 dataset):

2. **Viewing Image.** You can use the imageMagick tools to view the image (this is already installed in the VM). On a terminal, you can run the following command:

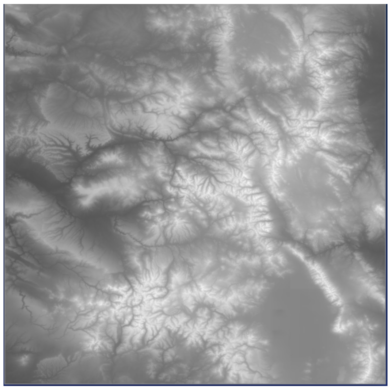
   - To directly view the ppm image, type

     display imagefile

     where imagefile is the image.
   - You can convert the image to ppm and view it. For this, type

     convert image.ppm image.png

     which will convert the file to a png image.



3. **Compute the Lowest Elevation Path.** Run your greedy algorithm on the image, output path to Image. The user will input the coordinates of a pixel (choose a pixel in the left most column) and your program will determine the points in the path that exhibit the smallest change in elevation and draw this path in a distinct color (like red). Pixels in the path will have their values changed to this color (for instance, use (MaxVal, 0, 0) for red. (See examples of image output in the nifty assignments website)

Requirements, Evaluation:

- You can implement this project as a purely procedural program, similar to project 1.
- You will use a 1D integer array to hold the image data, all addressing in the greedy algorithm will use 1D addresses; this will be used to create the input image and the final image with the path drawn on the elevation map.
- You will use functions/procedures to (1) read the dataset, (2) write it into PPM format, and (3) run the greedy algorithm.
- **Input/Output.** Your program will take as input a row number (column is assumed to be 0) and an input file name; and it will output the original image and the image with the elevation path drawn.
- Turn in your source code to Canvas and a sample output by the deadline; **Source code must be well documented for full credit.**