Pseudocode for the path searching for the mountain paths assignment.

/* As a note, at this point in the program, there are four vectors. One for original heights data, and three for grayscale data, red green and blue respectively. The red green and blue data has been converted on a 0 to 255 scale to represent pixels in an image in .ppm format. */

// The following nested loop will be to fill in the paths, and populate a vector with the distance //that each path takes overall.

Initialize a vector of size being equivalent to rows (the number of starting paths)
 //vector<int> length (rows)

(for) iterating from i = 0 to less than rows, add 1 to i each iteration{
  (for) iterating from j = 0 to less than col, add 1 to j each iteration{
  Int passed = 0;
  Int rowId = i; //saves the i.d. for the row without changing the loop value for each iter
  length.at(i) = 0;

  while(passed is less than the amount of columns){
    direction += checkNext(rowId, j, data)
      //checkNext() returns +1 for up, 0 for no change, and -1 for down
    length.at(i) += elevChange(j+1, rowId, data, direction)
      //above code assumes always going forward. J +1 is equivalent to the next column.
      //elevChange returns the absolute value for the change in elevation given the next square.
    rowId += direction
    changeColor(j+1, rowId, red, green, blue)
      //changes the color for the pixel being passed into
    add 1 to variable passed
      //after this, loops again until all columns have been reached
  }
} //ends the first for loop
} //ends the second for loop

//after the above sequence runs, the lowest value in the vector length is the shortest path. The address of the lowest //value will be the the starting row for the shortest path.

//The below pseudocode is just a search for the lowest total elevation change

Int smallest = length.at(0)
Address = 0
(for) i = 1 to less than the size of the vector “length”
  If the size of the value in length.at(i) is greater than the value for the integer smallest
    Smallest = length.at(i) //the new value for the shortest path is input
    Address = i //the starting row for the shortest path

//after this for loop, address should be equal to the starting row for the shortest path.
//after that, we will have to fill in the shortest path with green using a for loop

(for) i = 0 to less than the number for col (columns){

```
    direction += checkNext()
          //checkNext() returns +1 for up, 0 for no change, and -1 for down
    address += direction
    changeColor(i+1,address , red, green, blue, true)
    //true is added to change the default parameter, changing the color to green rather than red
    //address now holds the row value for the next step
}

//After that for loop, the shortest path should be colored in green.

////////////////////////////////////////////////////////////////////////////////////////////////////////////
Functions pseudocode

Int checkNext(int row, int  col, vector<int> data)  {
  Int i = -1;
  Int change = -1;

  Smallest = data.at(row+ i).at(col + 1)
  while( i < 2){
   If data.at(row+ i).at(col+1) < smallest{
     Smallest = data.at(row + i).at(col + 1)
     Change = i;
   }
  }
Return change;          // this will return if it is the next row above, on the same like, or below.
}

Int elevChange(int col, int row, vector<int> data, dir){
      Return abs(data.at(row).at(col) - data.at(row + dir).at(col+1)) //gives abs. val of the change in elevation
}

changeColor(int col, int row,vector<int>& red, vector<int>& green,vector<int>& blue, bool green = FALSE){
    if(!green)       //that is, if the function doesn't call for green
        Change the color values for red, green, blue.at(row).at(col) to RED
    Else
        Change the color values for red, green, blue.at(row).at(col) to GREEN
       //above case is ONLY for the shortest path case.
}
```