Pseudocode for the Mountain Paths algorithms.

```
Main function(){
Declare a vector of vectors for “Mountain Path Data” //To hold original elevation values.
Declare a vector of vectors for “RGB Scale Path Data”  //To hold RGB translation values
Declare an integer minimum
Declare an integer maximum
Declare a   double scale

Open a filestream to file “mountain.dat” //Can be this, or the name of any data file.

Check if the filestream was successfully opened.
        If the filestream was opened, continue.
        Else, if the filestream was not opened, supply the user with an error message.

//Begin reading data
While there is still data to be read (‘i’ value starts at 0, as well as ‘j’ value)
        Push back the data to the vector for “Mountain Path Data” at the “i”th row, “j”th column
        Add 1 to the value of i
        If i = 843 //because the index started at 0
                Set i to 0 again.
                Add 1 to the value of j
//After the while/for loop, the data should be correctly pushed back to the vector.
//For the next portion of this program, after this, there will be an algorithm to paste the vector //into two
other vectors for RGB, to manipulate after finding the correct path.

//Find the minimum and maximum values of the data in the vector
Set minimum value equal to the data in the first row of the first column of the data vector
Set maximum value equal to the data in the first row of the first column of the data vector
From index i = 0 to index i = the number of rows - 1
        From index j = 0 to index j = the number of columns - 1
                If the index at row i, column j is less than the index at row(i + 1) column(j + 1)
                        Set the minimum value to the data at row(i + 1) column (j + 1)
                Else if the index at row i, column j is greater than the index at row(i+1) column(j+1)
                        Set the maximum value to the data at row(i + 1) column (j + 1)
//This process can be used as a method or a standalone loop within the program.
//Now we set our values to a scale of 255

From index i = 0 to index i = the number of rows - 1
        From index j = 0 to index j = the number of columns - 1
                Data at row i, column j = ((data  - minimum) / (maximum - minimum)) * 255
                // With this code, we would be pushing these values into the RGB vector(s).

//Now that we’ve got the values in our vector(s), we can output them in a file.

Open an output stream called “data.ppm” or some other name.
```

```
From index i = 0 to index i = the number of rows - 1
        From index j = 0 to index j = the number of columns - 1
                Output the value from the RGB vector(s) at row i, column j to the file location.
                //Note: If one vector, repeat this three times, but if you have a vector for each RGB, do
                // this once for each vector in sequence.

End processing by returning 0.
```