



Problem

NASA has just hired you to code a program that will partially process “digitized” elevation grid pictures, (gray scale), of Titan, one of the moon’s of Saturn, received from the Hubble Telescope. The program first checks the grid for *feature errors*, correcting any that are found. The *peaks* and *valleys*, are then located and a chart is output denoting their locations. Lastly, the program determines if the region denoted by the grid is *mountainous*.

Discussion

The gray scale elevation picture will be represented as a two-dimensional matrix of values. Each value in the matrix represents a *pixel* or picture element. The pixel values range from completely white, 0 .. 99, to completely black. The picture matrix is stored in the files “**regionx.dat**”, with the format described below in the input section.

The program will make two *passes* through the matrix. The first pass will check for and correct transmission, (*feature*), errors that occurred during the transmission which causes feature errors in the picture. The second pass will find and store the location of the *peaks* and *valleys* in the picture.

Execution

The program should initially present the user with a startup screen displaying the name of the program, a brief 3-4 line explanation of the program and the programmer’s name and email address. After waiting until the user hits the return key the startup screen is to be cleared. A brief help screen should appear explaining the program to the user, which is cleared upon the pressing of the return key.

Following the brief help screen the user is prompted for the input and output file names and to wait while the picture matrix is being read. The user should be notified once input is complete and requested to wait again while the program checks for transmission, a.k.a. *feature errors*. A feature error exists if a location in the grid, (not on the boundary), differs from the mean of its eight surrounding *neighbor* locations by more than eight. If feature errors are found the row & column indices of each feature error should be output along with the impossible value. The error, (determined from the original matrix values only), should be corrected by replacing the incorrect value with the mean of its neighbors minus one. At the end of correcting all of the feature errors, the modified grid and its index number should be displayed. If no feature errors are found the user must be informed of this fact. The boundary is defined to be the first and last rows and columns.

Once the feature errors have been corrected, the user is again asked to wait while the program locates all of the *peaks* and *valleys* in the grid. A peak is defined to be any interior grid location, (this excludes the border pixels), which is greater than or equal to 5 or more of its neighbors. A valley is defined similarly, except the relationship is less than or equal to 5 or more of its neighbors. Once the *peaks* and *valleys* have all been located the user is notified and informed that a simplistic topol map will be output shortly once the program checks to see if the region is mountainous. A region is determined to be mountainous if 75% or more of its interior locations are either valleys or peaks. Once this determination has been made the program will output the corresponding topol map, (described in the output section), for the matrix. The user is then queried to determine if they wish to process further pictures. Before halting the program must issue a brief termination message before returning to the operating system.

Input

The input files, "**regionx.dat**", contains on line 1 the picture index number of the grid (a four digit nonnegative integer). Following the index number, (used internally by NASA for archiving), also on line one will be the dimensions (row x column) of the grid (maximum 25 x 25) separated by one or more spaces. On the following lines, (row number of them), of the file will be the values, (column number of them on each line), for the grid in the range [0 , 99]. The file must be checked to ensure that it contains the correct number of rows & column values in the corresponding matrix format as given on line one by the dimensions. If a file does not contain the correct number of row and column values, the offending row or column should be identified to the user in an error message and processing of the file is terminated. Input files will be made available shortly, downloading instructions will be given on the course web site: <http://ei.cs.vt.edu/~cs1344>.

Assumptions

It may be assumed that the input file will contain only integer values. Invalid range input must be checked, (< 0 | > 99), and appropriate error messages issued. If an input file contains range errors the program should locate & list all of the errors, and perform no picture processing on an unacceptable picture. Operating System error messages are unacceptable.

Output

A very simplistic topology map in the following format is to be output:

```

      Topology Map of Peaks & Valleys
      1      2      3      4      5      6
1  |-----|
2  |          ^          v
3  |          v          ^
4  |          |          |
   |-----|
      Legend:  ^ (peak)   v (valley)
```

All output, except user prompts, is to be echoed to both the screen and the user specified text output file, "**regionx.out**". Note that this does not require duplicating write statements if file redirection through C/C++ file variables is implemented.

Grading

Turn in hard copies of the source code, input/output files and a top-down hierarchical design tree of the program. In addition, submit a DOS (FAT) 3 1/2 HDD (1.4MB) diskette, with files containing: ASCII source code, executable image, I/O files, and an ASCII *readme* file with execution instructions. The source code file must be named topol.cpp and the executable image topol. The disk must contain no subdirectories. All materials to be graded must be placed in a sealed folder, neatly labeled, with your name, course number, program number and date.

You may be required to demonstrate your program. To receive partial credit for programs that are non-working, or are not fully functional, a brief one or two paragraph description of the problem(s) must be included in the assignment folder. The location, routine minimum, must also be specified along with possible corrections that need to be made.