CUDA Coding Competition Problem Statement

Introduction

This document describes the computations steps for the CUDA Coding Competition. Accompanying this document you will find example input files (input.h, input_large.h) and a script to build and run your code (submission.pl). Please rename input_large.h to input.h for testing large data-set.

Please ensure that your code runs successfully with the submission.pl script before submitting your code.

Section A - Terminology

1. Map:
   A "Map" is a 2-dimensional array of unsigned integers where each element represents the altitude of a cell.

   a. A cell is a HILL if all its 8 adjacent cells have a lower altitude.
   b. A cell is a DALE if all its 8 adjacent cells have a higher altitude.
   c. There can be no hill or dale on boundary cells.

   The example below shows a map with one Hill (in green) and one Dale (in blue).

5	5	5	5	5
5	7	5	3	5
5	5	5	5	5

2. Connected Regions:
   A connected region is a maximal set of adjacent non-zero values. All connected regions combined contain all non-zero values. The example below shows a map with three connected regions.

1	1	0	0	0	1	1
0	1	0	1	1	0	0
1	0	0	1	1	0	1
1	0	0	0	0	0	1
0	1	0	1	1	1	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0

3. Mean of surrounding cells:

   The mean of the surrounding cells is the arithmetic average of the surrounding cells, computed by summing all the values and then dividing by the count of cells. For the map below, we can compute the mean of the cells surrounding X as follows:

A	B	C
D	X	E
F	G	H

$$Mean(X) = \frac{(A + B + C + D + E + F + G + H)}{8}$$

4. Median of surrounding cells:

   The median of the surrounding cells is defined as the arithmetic average of the middle values in the sorted list of values. For the map below, we can compute the median of the cells surrounding X as follows:

A	B	C
D	X	E
F	G	H

Compute Median as:

Sort the surrounding cells and

$$Median(X) = \frac{(Value\ at\ 4th\ location + Value\ at\ 5th\ location\ )}{2}$$

Section B – The Challenge

- **Step 1**
  - For the given input set which contains N maps, find all possible Hills and Dales based on the definitions in section A, point 1. Note that the input may contain multiple maps and each map may be of a different size. See section D for examples.

    The program should #include the header file "input.h". This file contains the function get_input() which returns a pointer to an array of unsigned integers.

Your program needs to read input from this array; you can assume the input is error free. The output must be written to STDOUT in the format described below section C - Output

Note that when you have submitted your code the file "input.h" will be replaced with a test data set for confirming correctness and performance. You will not have access to the test data set.

  - For all maps:
    - Replace all Hills with the mean value of the surrounding cells as defined above in section A, point 3.
    - Replace all Dales with median value of the surrounding cells as defined above in section A, point 4.

- **Step 2**
  - For all maps, repeat *Step 1* until either no Hills and Dales remain in the map or the number of iterations (including the first iteration in *Step 1*) reaches k. The value of k is defined in "input.h" as NUM_ITERATIONS.
  - Keep in mind that at the end of each iteration a different map will be yielded and should be taken as the input for next iteration. For details, see the example below in Section D.

- **Step 3**
  - Convert Maps from *Step 2* to binary Maps as follows:
    - Compute the average value of a given Map by computing the average of all cells. This is the "Threshold", T. The average will be the sum of all cell values divided by total number of cells, rounded down to the smaller integer.
    - For all cells in the map, assign labels as follows:
      - If (Cell Value < T) Cell Value = 0;
      - Else Cell Value = 1;

- **Step 4**
  - Count the number of connected components in all binary Maps from *Step 3*. To learn more about connected components please see link http://en.wikipedia.org/wiki/Connected-component_labeling
  - The output will be the Map number followed by the corresponding number of connected components in that Map. See the example output in section D below for more details.
  - We will take 8-connected objects (meaning that diagonally adjacent 1s are considered connected).
  - Please note during evaluation only the output of last step will be verified, there is no need to print output for the intermediate steps and doing so may cause your submission to fail verification. The evaluation will verify the Map number and

corresponding number of connected components in that map. See output in section D below for more details.

Section C – Input and Output

Input

The input data array will be in the following format: the first element of input data array i.e. input[0] - contains the number of Maps, N. N Maps will follow, each starting with two integers -- R and C -- the number of rows and number of columns of the map respectively. The next C elements will represent the first row of the map, from West to East. R such rows will follow. Each data element of the map represents the altitude of the cell, A.

Output

For each map output a line in the format:
MAP #M: count
M is the map number starting from 1. count is the number of connected objects found in the map M.

Limits

N, R, C and A are all unsigned integers bounded by following limits.
1 <= N <= 100
3 <= R, C <= 16,384
0 <= A <= 65,535

Section D - Sample Input file and Example

D.1 Example Input

NOTE: The actual input used for evaluation will be generated dynamically. You must call get_input() in your program to get a pointer to the input data.

Contents of sample header file "input.h"

```
unsigned int *get_input(void)
{
  static unsigned int input[] = {
    3, // # maps
    // Map 1
    3, 3, // # rows, #cols
    5, 5, 5, // row 1
    5, 5, 7, // row 2
    5, 5, 5, // row 3
    // Map 2
    3, 5, // # rows, #cols
    5, 5, 5, 5, 5, // row 1
    5, 7, 5, 3, 5, // row 2
    5, 5, 5, 5, 5, // row 3
    // Map 3
    5, 3, // # rows, #cols
    7, 7, 7, // row 1
    7, 8, 7, // row 2
    7, 8, 7, // row 3
    7, 6, 7, // row 4
    7, 7, 70 // row 5
  };
  return input;
}
```

We can represent this data pictorially as below:

MAP 1

5	5	5
5	5	7
5	5	5

MAP 2

5	5	5	5	5
5	7	5	3	5
5	5	5	5	5

MAP 3

7	7	7
7	8	7
7	8	7
7	6	7
7	7	70

D.2 Example Steps

The following steps will illustrate the intermediate results from each step in Section B.

Please ensure that your code runs successfully with the submission.pl script provided with this problem statement before submitting your code.

Step 1

- For all maps, identify the Hills and Dales as highlighted below

  Remarks
  Map 1: 7 is not a hill (section A condition 1c)
  Map 2: 7 is hill (section A condition 1a), 3 is dale (section A condition 1b)
  Map 3: 6 is a dale, 8s are not hills (do not match section A condition 1a), 70 is not a hill (section A condition 1c)

MAP 1

5	5	5
5	5	7
5	5	5

🟦	Dale
🟩	Hill

MAP 2

5	5	5	5	5
5	🟩 7	5	🟦 3	5
5	5	5	5	5

MAP 3

7	7	7
7	8	7
7	8	7
7	🟦 6	7
7	7	70

- Modify the maps to replace Hills with the mean of the surrounding cells and Dales with the median of the surrounding cells. Modified Hills and Dales are highlighted.

MAP 1

5	5	5
5	5	7
5	5	5

MAP 2

5	5	5	5	5
5	5	5	5	5
5	5	5	5	5

MAP 3

7	7	7
7	8	7
7	8	7
7	7	7
7	7	70

Step 2

Iterate *Step 1* for either k number of iteration or no Hills and Dales left in map.

In Iteration 2, we see there are no Hills and Dales here on the output maps from Step 1 (according to the definition in section A). As there are no Hills and Dales left, we will stop iterations here.

Step 3

- Compute Threshold for each map

MAP Number	Threshold
1	47/9= 5
2	75/15=5
3	170/15= 11

- Create the binary maps

MAP 1

1	1	1
1	1	1
1	1	1

MAP 2

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

MAP 3

0	0	0
0	0	0
0	0	0
0	0	0
0	0	1

D.3 Example output

The final output for the above example would look like this:

```
Map #1: 1
Map #2: 1
Map #3: 1
```