Middle East Technical University Department of Computer Engineering

CENG 140

C Programming

Fall '2018-2019

Take Home Exam I

Due date: November 30th, Friday, 23:55.

1 Recursive Part - Wells of the Mountain

In a warm and calm country far away, there is a huge mountain that surprisingly attracts most of the clouds around it, causing a great amount of spring water spots residing several places of the mountain. Around each water spot, there is a small village and at the top of each spot there is a well built by the people of that village.

The water does not disappear immediately after it comes out of the well. It pours through the villages located in lower territories of the mountain, breeding the farms in more than one villages and makes the villagers, especially inhabited in the lower altitudes, more content than upper ground villagers.

Some day a wise man visits peaceful people of giant mountain. He travels most of the villages and sees the wells of the villages and a question begins to emerge in his minds: "Can I measure the happiness of farmers in this place?" He knows that if the people of the mighty mountain is in question, this can be possible only if he knows how much water arrives to their village. Therefore, he start to explore the mountain and note down some significant findings. He realizes that the wells, together with their villages, are located all over the mountain in a strict shape: a triangle. There is one well at the peak of the mountain and right below it there are two wells residing lower right and left of that well and having the same height. Below the second level there are three wells at the same height and having equal distance apart from each other on a straight line. He draws the sketch of well formation in Figure 1.

He discovers that each well has unlimited and constant water pouring 100 liter per hour. Maybe the most important observation he get is water paths going to lower villages. The water of the well, together with the water coming from upper wells forks to two identical streams arriving to left down and right down villages. For example, in Figure 1, he drew that the water coming to village E combines with the water coming out of well of village E is split into streams and pours to village H and village I with equal amounts.

He manages to calculate the water accumulated on some lower villages. For instance, he can say that village B receives 50 lt. per hour from Village A only. Village H takes:

- Half of the stream leaving Village D (87.5 lt) which is

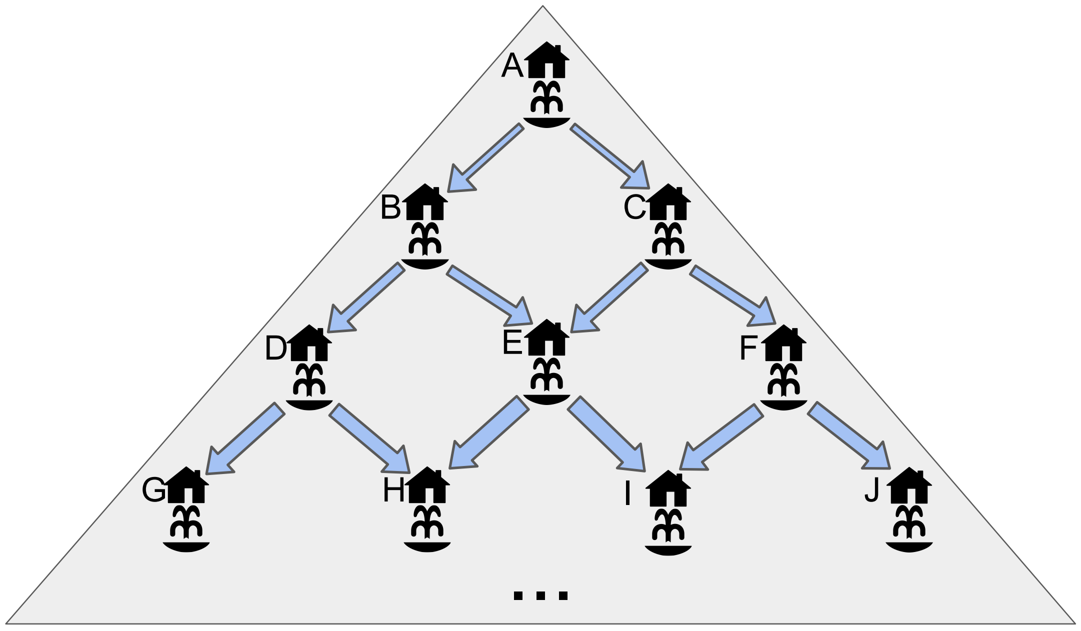

Figure 1: Wise man's sketch.

  - Half of stream leaving Village B: (50 +100)/2 = 75 lt.
  - The water supply of its own well: 100 lt.
- Half of the stream leaving Village E (125 lt) which is
  - Half of stream leaving Village B: (50 +100)/2 = 75 lt.
  - Half of stream leaving Village C: (50 +100)/2 = 75 lt.
  - The water supply of its own well: 100 lt.

Calculating the water coming to the Village H, which is 125 + 87.5 = 212.5 lt, makes him think that the villagers of H is more happier than Village B and Village E. However, he instantly remembers that there are so many villages on this giant mountain and his lifetime is limited like every mortal creature in the world. Then he takes his laptop out of its case (since he is a wise man of modern times) and begins to write a C code for calculating the total amount of water that coming to a village similar to the one you are expected to implement in this part of the Take Home Exam.

Specifications

In this section, you are expected to solve the problem specified in the previous part. You will implement a recursive solution with the given specifications:

- You will be given the location of the village from standard input. Your program will print the amount of water that accumulated on that village. **The water from the well of that village will not be added to the result.** For example, if Village A is queried in the input, the result will be 0. For village B, the result will be 50 etc.

- The location of a village is defined as <**the level of village, the horizontal index of village**> both of which has type `integer`.
- The level indicates the height of a village. Levels start from 0 and they are numbered in increasing order as the height goes down. For instance, levels of Village A, B, and H are 0, 1, and 4, respectively.
- The horizontal index of village is the index of the village among the villages in the same level. It starts from the leftmost village in a level and its starting index is zero. For example, The horizontal index of A, E, and J are 0, 1, and 3, respectively.
- Input-output examples are given as follows:

```
Input 1:
1 0

Output 1:
50.000000

Input 2:
3 1

Output 2:
212.500000
```

- The output should be printed to standard output as `double`. Do not forget to put a new line character at the end.
- You can assume that integers greater than 100 will not be used as inputs.
- Decreasing the time complexity of your implementation is recommended but not mandatory.
- Do not use static or global variables.
- For collecting credits in this part, avoid use of iterative statements.

2 Iterative Part - Path to Pure Happiness

Our modern wise man, like any other wise man lived so far, is apt to pursue answers to big questions. Putting the effort to the solution in the first section makes a related question arise in our man's mind: Where does the path of *pure* happiness pass in this mountain? Not surprisingly, the answer has to be associated to water. However, this time he has a different point of view and comes up with the statement: "The water which passes through the villages with less population is the purest." He interprets the statement that he should find the water path passing through the villages which is the least populated in total. Similar to the previous search in Section 1, he advised this problem to his best friend, his laptop, by implementing a solution in C language. Nevertheless, he has no idea that his task is exactly the second section of the first Ceng140 Take Home Exam.

Specifications

In this section, you are expected to solve the problem specified in the previous part. You will implement an iterative solution with the given specifications:

- You will be given the number of levels and the village populations of the entire mountain from standard input. Your program will basically print a path from the peak of the mountain to a village residing in the lowest level of the mountain. This path should have the following property: the sum of village population on this path has the lowest value among all the paths from the top to the bottom of the mountain.
- Path directions between two consecutive village is the same as water streams described in Section 1 In other words, as the next step, **you can only move lower right or lower left village**. Moving left or right in the same level or going upper villages is not possible. Only traveling to lower level is allowed and it is restricted with going immediate left low and immediate right low villages.
- Input-output examples are given as follows:

```
Input 1:
5
134
39 268
311 610 50
647 405 103 516
278 156 809 262 456

Output 1:
134 268 50 103 262

Input 2:
7
12
24 43
44 456 54
647 12 103 516
545 156 9 348 4
671 192 89 513 473 9
725 56 40 38 5 7 2

Output 2:
12 24 44 12 9 89 38
```

- The output should be printed to standard output. Do not forget to put a new line character at the end.
- Both input and output values are in `integer` types.
- You can assume that the input sequence with maximum level more than 100 will not be tested.

- Decreasing the time complexity of your implementation is recommended but not mandatory.
- Do not use static or global variables.
- For collecting credits in this part, avoid use of recursion.

3 Compile and Run

- Both solutions will be compiled "gcc" containing the flags enforcing ANSI standards: `-ansi -pedantic-errors -Wall`
- Obey I/O format strictly. Otherwise you will not be able to collect points since the exam will be graded by blackbox technique.

4 Deliverables

- Your source files should be named as `hw1_recursive.c` for recursive implementation and `hw1_iterative.c` for iterative implementation.
- You are expected to submit both files as a compressed file named `hw1.zip` to COW course page before the deadline. The compressed file should not contain any directories, it should only contain 2 source files.
- Follow newsgroup and course page on COW for further announcements and possible updates on a daily basis.