Middle East Technical University Department of Computer Engineering

CENG 140

C Programming

Spring'2022-2023
Take-Home Exam 1

Yusuf Mücahit Çetinkaya
yusufc@ceng.metu.edu.tr
Due date: May 1, 2023, Sunday, 23:59

1 Overview

Recursion is a concept in computer science in which a function calls itself by directly or indirectly breaking the input into pieces to solve problems. These functions are called recursive functions. The recursion method perfectly fits some problems, such as Towers of Hanoi, tree traversals, graph searches, etc. However, sometimes iterative solutions are more straightforward and more appropriate.

In this assignment, you are expected to implement some functions with iterative (Part 2.1) and recursive (Part 2.2) approaches. Signature of the function for Part 2.2 is given in the corresponding task file.

2 Tasks

2.1 D'Hondt method (40 pts.)

Elections are a fundamental aspect of democratic societies, allowing citizens to choose their representatives and have a say in how their government is run. In many countries, elections use a proportional representation system to allocate seats in parliament or other legislative bodies. The D'Hondt method is one such proportional representation method widely used worldwide. It aims to allocate seats to parties in proportion to the number of votes they receive based on a formula that calculates the quotient for each party. The party with the highest quotient is allocated a seat, and the process continues until all seats are allocated.

In this task, you will implement a program that calculates the number of seats each party won using the D'Hondt method with an iterative solution. Your program should read the inputs from standard input and print the results to standard output. Your program should satisfy the following requirements:

- The input format should be as follows:
  - The first line should contain two space-separated non-negative integers, N and M, representing the number of parties and the number of deputies, respectively, where $N \leq 50$ and $M \leq 100$ inequalities hold.
  - The second line should contain N space-separated integers, representing the number of votes for each party in the order A, B, C, ..., Z.
  - You can assume that the input values are valid, i.e., the number of parties and deputies are positive integers, and the number of votes for each party is different and a non-negative integer.
- The output format should be as follows:
  - The program should print N lines for each party in descending order of the number of seats won.
  - Each line should contain the party name (A, B, C, ..., Z) followed by a colon and a space, then the number of seats won by the party.
  - Parties not getting any seat should not be printed out.
- The program should use the D'Hondt method to allocate seats to each party based on the number of votes. The D'Hondt method works as follows:
  - Calculate the quotient for each party by dividing the number of votes for that party by the number of seats it currently holds plus one.
  - Allocate a seat to the party with the highest quotient.
  - Repeat the above steps until all seats have been allocated.

- If two or more parties have equal quotients after dividing the number of votes by the seats they currently hold plus one, then the D'Hondt method prescribes that the party with the highest number of votes should be given the seat.
- **Input-Output example:** You will implement a main function that gets the inputs from stdin and prints to stdout.

  Input:

  ```
  4 10
  100 200 300 25
  ```

  Output:

  ```
  C: 5
  B: 3
  A: 2
  ```

  In this example, there are 4 parties (A, B, C, D) and 10 deputies. Party A has 100 votes, party B has 200, party C has 300, and party D has 25. The D'Hondt method is used to allocate seats to each party. Party C gets the first seat with the highest quotient (300 / 1 = 300). While calculating the second deputy, Party A, B, and D votes will be divided by 1 but Party C's vote will be divided by 2 since it will be the second seat for Party C but first for others. Party B gets the second deputy with the highest quotient after allocating the first seat (200 / 1 = 200). Party C gets the third seat with the highest quotient after allocating the second seat (300 / 2 = 150). The process continues until all seats have been allocated. The final result is that party C gets 5 seats, party B gets 3, party A gets 2, and party D gets none. Please apply D'Hondt method as described here, alternative descriptions on the Web may confuse you and cause losing points.

2.2 Egyptian pyramid pressure (60 pts.)

The Egyptian pyramid is a marvel of engineering and architecture that has captured the imagination of people for centuries. These monumental structures were built around 4,500 years ago in ancient Egypt's Old Kingdom period. One of the fascinating aspects of the pyramids is how the enormous limestone blocks were placed on top of each other to form the iconic shape. Each block was carefully cut and fitted into place with remarkable accuracy, and what is even more impressive is the fact that there was virtually no space between the blocks. This means that the pressure exerted by each block on the one below it is immense, yet they have stood the test of time and remain standing today. The sheer scale of the pyramids and the precision with which they were built is a testament to the skill and ingenuity of the ancient Egyptians.

In this task, you will develop a program that prints pressures on the blocks of an Egyptian pyramid, with 8 digits after the comma, by implementing **recursive** function. **Implementations that include iterative approaches won't get any grade.** Your program should read the inputs from standard input and print the results to standard output. Your program should satisfy the following requirements:

- The input format should be as follows:
  - The first line should contain two space-separated non-negative integers, N and W, representing the number of layers and the weight of each block, respectively.
  - $3 \leq N \leq 50$ and $10 \leq W \leq 100,000$ inequalities will be hold.
- The output format should be as follows:
  - The program should print N lines, one for each block, in ascending order of the levels and left-to-right order.
  - Each line should contain the block coordinate and its pressure.
  - Pressures will be displayed with 8 decimal points.
- Each block is represented with a coordinate of (x, y) where x is the layer's number begins with 0 starting from the top and y is the position of the block in the row where the leftmost block is counted as (x, 0) as shown in Figure 1.

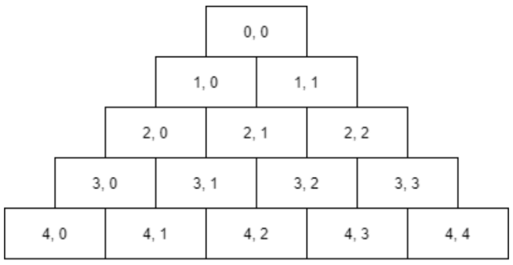

Figure 1: Indexes of the pyramid blocks with an example of 5 layers.

- Every block transfers half of its weight to the lower left and lower right blocks. Also, every block transfers 15% of the pressure on itself to the lower left and right blocks.
- An example of the pyramid is given in Figure 2.

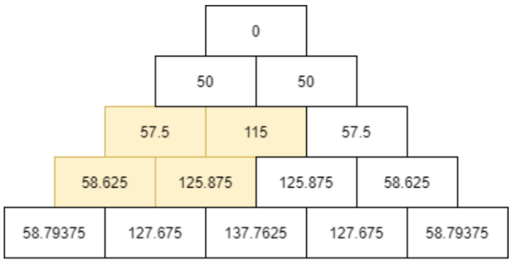

Figure 2: An example of pyramid where $N = 5$ and $W = 100$. The pressure on the blocks is shown on each of them.

- A sample calculation of the highlighted cells is visualized in Figure 3.

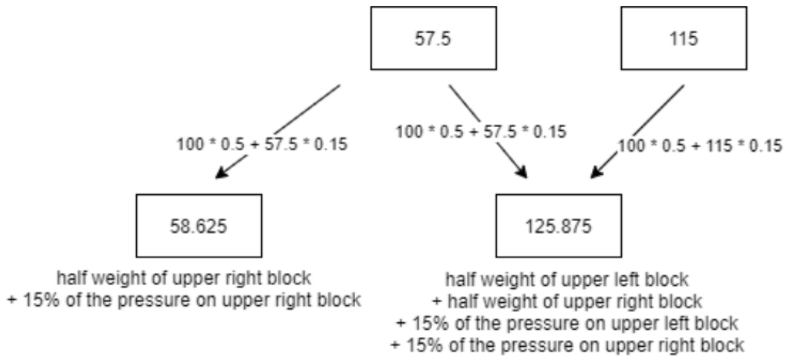
57.5
115
100 * 0.5 + 57.5 * 0.15
100 * 0.5 + 57.5 * 0.15
100 * 0.5 + 115 * 0.15
58.625
125.875
half weight of upper right block
+ 15% of the pressure on upper right block
half weight of upper left block
+ half weight of upper right block
+ 15% of the pressure on upper left block
+ 15% of the pressure on upper right block

Figure 3: An example of pyramid where $N = 5$ and $W = 100$.

- The recursive functions must not include any loop or global, static variables.
- **Input-Output example:** You will implement a main function that gets the inputs from stdin and prints to stdout and recursive functions that calculate the pressure on cells.

  Input:

  ```
  5 100
  ```

  Output:

  ```
  (0,0) 0.00000000
  (1,0) 50.00000000
  (1,1) 50.00000000
  (2,0) 57.50000000
  (2,1) 115.00000000
  (2,2) 57.50000000
  (3,0) 58.62500000
  (3,1) 125.87500000
  (3,2) 125.87500000
  (3,3) 58.62500000
  (4,0) 58.79375000
  ```

```
(4,1) 127.67500000
(4,2) 137.76250000
(4,3) 127.67500000
(4,4) 58.79375000
```

3 Regulations

- **Programming Language:** C
- **Libraries and Language Elements:**
  You should not use any library other than *"stdio.h"* and *"math.h"*. You can use conditional clauses (switch/if/else if/else), loops (for/while). **You can NOT use any further elements beyond that (this is for students who repeat the course).** Using any high-level data structure (pointers, structs, etc.) is strictly **forbidden**. You can define your own helper functions.
- **Compiling and running:**
  DO NOT FORGET! YOU WILL USE ANSI-C STANDARTS. You should be able to compile your codes and run your program with given **Makefile**:

  ```
  >_ make the1_dhondt
  >_ ./the1_dhondt
  ```

  and

  ```
  >_ make the1_pyramid
  >_ ./the1_pyramid
  ```

  If you are working with ineks or you are working on Ubuntu OS, you can feed your program with input files instead of typing inputs. This gives the input from stdin and an equivalent of typing inputs. This way you can test your inputs faster:

  ```
  >_ ./the1_dhondt < inp_dhondt.txt
  >_ ./the1_pyramid < inp_pyramid.txt
  ```

  If you are working on OdtuClass VPL, you can directly run your code and evaluate it to see your grade on sample test cases. Don't forget, we will evaluate your homework with additional test cases after the submission.
- **Submission:**
  You will use OdtuClass system for the homework just like Lab Exams. You can use the system as editor or work locally and upload the source files. Late submission IS NOT allowed, it is not possible to extend the deadline and **please do not ask for any deadline extensions**.
- **Evaluation:** Your codes will be evaluated based on several input files including, but not limited to the test cases given to you as example. You can check your grade with sample test cases via OdtuClass system but do not forget it is not your final grade. Your output must give the exact output of the expected outputs. It is your responsibility to check the correctness of the output with the invisible characters. Otherwise, you can not get grade from that case. If your program gives correct outputs for all cases, you will get 100 points.
- **Cheating: We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations and will get 0. Sharing code between each other or using third party code is strictly forbidden. Even if you take a "part" of the code from somewhere/somebody else - this is also cheating. Please be aware that there are "very advanced tools" that detect if two codes are similar. So please do not think you can get away with by changing a code obtained from another source.