Middle East Technical University | Department of Computer Engineering

CENG 140

C Programming

Spring '2021-2022
Take-Home Exam 1

Due date: May 9, 2022, Monday, 23:59

1 Overview

The Great Wall of China is the longest defensive wall in the world, stretching along the northwest of China to defend the country's borders against the attack of the "Turkish" and "Mongol" tribes against China from the north, especially the Hiung-nu (Great Hun Empire). Its ruins begin on the seashore in Po Hay bay. Passing north of Beijing, it heads west and bisects the Huang-Ho River, stretching to the southwest. It continues from the south of the Gobi Desert, heading west. The length of the Great Wall, with its destroyed parts, is 8851.8 kilometers. The part that stands today is the 2,500-kilometer embankment from the Ming Dynasty. However, the actual construction was done between 221 BC and 608 AD. According to another archaeological study, it has a total length of 21,196 km with all its branches.

In this assignment, you are expected to implement a simulation of strengthening the wall after the attacks of the tribes. You will submit a single file called *the1.c* for the homework.

2 Task

The wall is initially thin and China strengthens the parts where they get attack. Although, the Great Wall is not linear, we will assume that it is in the form of a simple line. The defender uses the following strategy to react to enemy tribes: whenever a part of the wall is attacked by an enemy of height 'h', the part of the wall in that region thinner than 'h' is strengthen to the thickness 'h'.

Tribes attack to the wall frequently. In this assignment, we assume that each enemy army has a width 'w' and height 'h' and attacks to the wall on some interval which equals to its width. In order to breach, enemies may continue to attack the wall in different places at different times by moving left or right with a constant speed. Note that, since the wall is tall enough, **enemies can't damage the wall**. However, after the attack, every attacked fragment of the wall that was thinner than 'h' is strengthen to a thickness of 'h' in order to repel possible future attacks to the same place.

Each tribe moves left or right at some constant speed but can only attack in some predefined constant time period. They attack the wall repeatedly with a fixed number of times. Assuming that initially the wall has a thickness of 1 unit at all places, and given the full description of all the tribes that attack the wall, the task is to determine the final state of the Great Wall after all tribe attacks are over.

The simulation will get the following parameters;

- r: the length of the wall, with $100 \leq r \leq 100000$,
- k: number of tribes, with $1 \leq k \leq 100$,
- w_i: width of the i^{th} tribe, with $1 \leq w_i$,
- h_i: height of the i^{th} tribe, with $h_i \leq 100$,

- t_i: the time of the first attack of the i^{th} tribe, with $1 \leq t_i \leq 1000$,
- p_i: the position of the left most position for the first attack of the i^{th} tribe, with $p_i \geq 0$ (effected part of the wall will be $[p_i, p_i + w_i + 1]$,
- s_i: the speed of the i^{th} tribe, with $|s_i| \leq 1000$ where negative values for going left and positive values for going right,
- a_i: attack interval of the i^{th} tribe, with $a_i \geq 1$,
- n_i: number of attacks carried by the i^{th} tribe, with $0 < n_i \leq 100$.

3 Regulations

- **Input-Output format:** You will provide a single file including main function with the name *the1.c*. The output should be a single line output of $r + 1$ integers separated by single spaces, indicating the latest thickness values of the Great Wall from position 0 to position r.

  Sample input:

```
20
2
3 4 2 5 1 2 2
5 2 5 10 -1 3 3
```

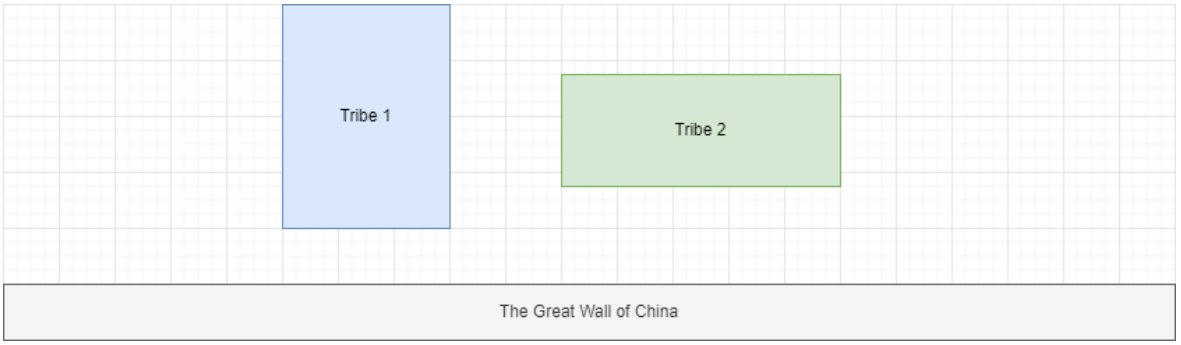

Figure 1: The initial state of the wall and the tribes.

Sample output:

```
1 1 1 1 2 4 4 4 4 4 2 2 2 2 2 1 1 1 1 1 1
```

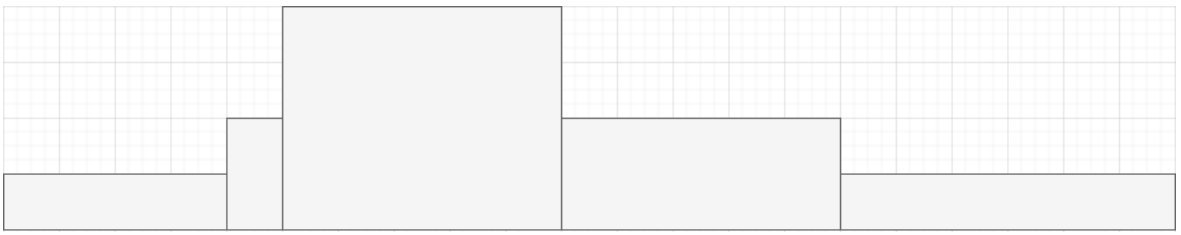

Figure 2: The final state of the wall.

Sample trace:
The first tribe attacks twice at minutes 2 and 4. The first attack hits the wall on positions 5, 6, 7. Then the tribe moves 2 units to the right in 2 time steps, with 1 unit/time step speed and makes its second attack at positions 7, 8, 9. After these attacks, the thickness of the Great Wall is strengthen to 4 units at these attacked positions. The second tribe attacks three times at time steps 5, 8, and 11. The first attack hits the interval [10,14]. The tribe moves left with speed -1 units/time step and the second attack hits the interval [7,11]. The last attack hits the interval [4,8].

- **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).** 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
>_ ./the1
```

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 < inp1.txt
>_ ./the1 < inp2.txt
```

- **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.