Project 2: HackNPaper

CMPE 250, Data Structures and Algorithms, Fall 2020

Instructor: H. B. Yılmaz

TAs: Berk Atıl, Rıza Özçelik

Due: **18.12.2020 Friday, 23.55**

1 Introduction

Heraclitus once said "The only constant in life is change". Heraclitus was a wise man, indeed, but he missed an exception: if you take CMPE 250, you solve a Discrete Event Simulation (DES) project!

Well, but what is DES?

Enter, Wikipedia: "A discrete-event simulation (DES) models the operation of a system as a (discrete) sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur; thus the simulation time can directly jump to the occurrence time of the next event". In other words, DES is the representation of a certain process by simulating certain events which occur at certain times. Hence, time is not continuous in DES and progress according to the events. As a demonstrative example, a process of airplane boarding can be simulated as follows:

- TIME: 09:00 Go into the luggage queue.
- TIME: 09:15 Give your luggage and take your boarding pass.
- TIME: 09:20 Go into the security queue.
- TIME: 09:30 Wait for your boarding time and enjoy your free time.
- TIME: 10:00 Go into the boarding queue.
- TIME: 10.30 Have a nice flight!

Note that in DES the only things that count are events and everything else is ignored. In the case of the passenger, until 10:31 only 6 things happened. Additionally, the time progressed in a discrete fashion (i.e. jumped from one event to the next one). [1]

[1] For further reading about DES you can visit here or Wikipedia

In this project, you are expected to simulate HackNPaper, which is a hackathon organized by Dunder Mifflin, Scranton.

2 HackNPaper

Dunder Mifflin is a paper company that operates in the Northeastern US. Their Scranton[2] branch is known for creative ideas such as the Dundies, fun run for rabies, and charity casino night for boy scouts. All thanks to their regional manager-extraordinaire: Micheal Scott.

Michael is an old-fashioned manager but his new superior, Ryan Howard, is determined to modernize the company. Ryan plans to launch a new website, Dunder Mifflin Infinity, and asks Michael to organize a hackathon to summon the best hackers together. Besides being the "world's best boss", Michael is a great "delegator" and delegates the organization to the party-planning committee led by Angela Martin. Angela is very serious at her job and ambitious to throw an excellent party. Ordered by Michael, she aims to maximize the number of hackers in HackNPaper and she knows exactly how to allure them: by giving away free stickers and hoodies! No developer can say no to that, right?

There is one "little" problem though. The party-planning committee is afraid that programmers might spend too much time in the sticker and hoodie queues that they will not focus on Dunder Mifflin Infinity. The committee can increase the number of sticker and hoodie desks to amend the problem, but this would cost extra money and Angela is quite stingy. **So, they decided to simulate the HackNPaper with fictional hackers to find how many desks would be enough**. This is where you will help them!

Here is a summary of everything related to HackNPaper. The hackers arrive at different times to the event and they regularly commit code snippets. They are also very much interested in the free stuff and visit sticker and hoodie desks which are placed sequentially. To grab the stickers and hoodies, one first visits the sticker desk and waits in the sticker queue until the desk is available. When the waiting is finally over, the hacker is served with the sticker and immediately enters the hoodie queue. Let Angela provide more details.

- There might be multiple sticker and hoodie desks, depending on the simulation configuration. In this case, every desk is operated by an attendant who serves the gifts (stickers and hoodies) and the service takes some time due to paper-works and stuff. Since not every attendant is equally fast, every desk has a different service time.

- There are exactly two queues in the event: one for stickers and one for hoodies. So, the sticker desks share **a common queue**, and so do the hoodie desks.

- The sticker queue works in first-come-first-served fashion. Thus, the first hacker to enter the queue is served before the others. If two hackers arrive at the same time, the one with the lower ID is served first. When the hacker is **served**, he/she enters the hoodie queue, **immediately**. Since this queuing system would be vulnerable to "free-riders", the attendants do not allow hackers with less than 3 commits to enter the queue and call such attempts "invalid".

- The number of commits is a vulnerable metric to cheaters as well... So, as an additional measure, the committee counts only the commits with line changes over 20 towards queue

[2] A video to get to know Dunder Mifflin, Scranton.

entrance. In other words, the commits with less than 20 changed lines are simply irrelevant to deserve the right to enter the sticker queue.

- Because hoodies are very popular and cute, the hackers must deserve them by showing some merit! Well, in the coding world, merit is measured by the number of commits. Therefore, in the hoodie queue, **the more the commits, the earlier the service**. If any two hackers have an equal number of commits, the one that arrived earlier is served before. If they arrived at the same time as well, the one with the lower ID is served first.
- Both for the sticker and hoodie desks, hackers visit the first available desk for service (the available desk with the smallest ID) when they leave the queue.
- Each hacker is allowed to take at most **3** stickers and hoodies (leave some for others, people!). Hence, whenever a hacker attempts to enter the queue 4^{th} time, this is also called an "invalid attempt".
- To make the hoodie queuing simpler, the commits are made only while waiting in a sticker queue or during the period out of the queues.
- One can enter the hoodie queue only through a sticker desk, no direct entrance is possible.
- Hackers are associated with increasing IDs as they arrive in HackNPaper. The first comer is assigned with ID 1, the next one is assigned with 2, etc.

Below, is a schematic description of HackNPaper.

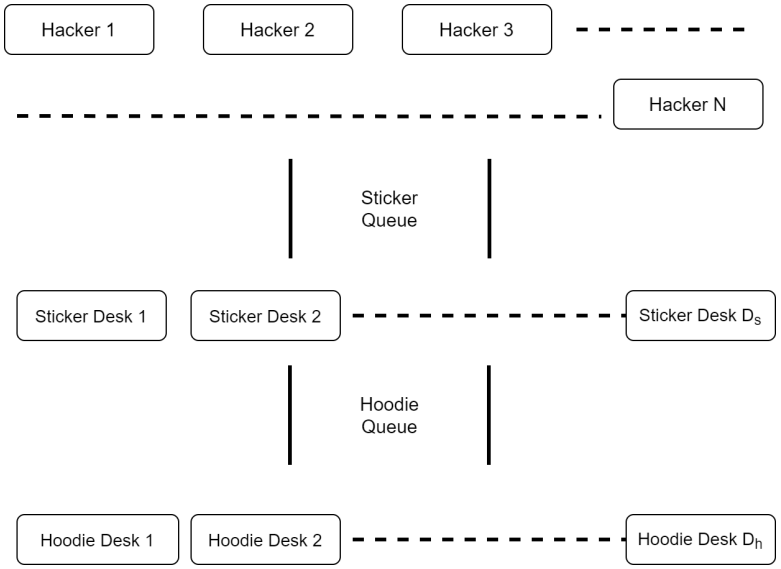

Figure 1: A schema of the queues

Therefore, for our purposes, there are three types of events triggered by the hackers: arrival, code commit, and queue entrance attempt. Keep also in mind that internal events, such as leaving the desk, should also be triggered by the simulation. The committee expects you to simulate HackNPaper using these external and internal discrete events and collect some statistics.

3 Input & Output

Your code must read the name of the input and output files from the command line. We will run your code as follows:

- g++ *.cpp *.h -std=c++11 -o project2
- ./project2 inputFile outputFile

Make sure that your final submission compiles and runs with these commands.

3.1 Input

The committee provides all simulation configuration files in the following format:

- The first line contains an integer N that denotes the total number of hackers and N floats that denote the arrival time of each hacker in terms of seconds passed from the beginning of HackNPaper.
- The second line contains an integer C which is the total number of code commits, including the ones shorter than 20 lines.
- Each of the next C lines contains two integers and one float: the ID of the hacker that commits the code, the number of line changes in the commit, and the second T in which the commit is made.
- The next line is the line of an integer Q that denotes the number of queue entrance attempts.
- Each of the the next Q lines contains one integer and one float which are the ID of the hacker and the second he/she attempts to enter the sticker queue, respectively.
- The next line comprises an integer D_s that denotes the number of sticker desks and a list of floats of size D_s. The i^{th} element of the list denotes the service time of the i^{th} sticker desk.
- The next line starts with an integer D_h and is followed by a list of floats of size D_h. Similar to previous line, the i^{th} element of the list denotes the service time of the i^{th} hoodie desk.

Table 1 demonstrates an example input provided by Pam Beesly, the lovely member of the committee.

3.2 Output

The committee needs you to collect the following statistics to evaluate the configuration and output them in separate lines, in the order they are given. Please round the statistics to **exactly 3 decimal points**. In case you cannot report any of the following 13 statistics, **print -2** for each of them in order to adhere the output format.

Sample Input File	Explanations
3 1 2 3	There are 3 Hackers and they arrive at second 1, 2 and 3
13	There are 13 code commit events
1 20 5.5	"Hacker 1" commits a code with length 20 at second "5.5"
1 21 6.3	
2 20 7.1	
3 30 8	
1 23 10	
2 25 11	
3 20 11.5	
3 21 15	
2 30 15	
3 25 17.5	
1 20 18.1	
2 40 18	
1 25 20	
8	There are 8 queue entrance events
1 9	Invalid attempt to go to a queue event due to insufficient code commits (2)
1 11	"Hacker 1" is going to a queue at second "11"
3 18	"Hacker 3" is going to a queue at seconds "18"
2 18.1	
1 18.5	
3 26	
3 38	
3 46	Invalid attempt to go to a queue event due to exceeding number of gifts (4)
1 3	1 desk for stickers and with service time of 3 seconds
1 4	1 desk for hoodies and with service time of 4 seconds

Table 1: Sample Input with Explanations

1. Maximum length of the sticker queue.
2. Maximum length of the hoodie queue.
3. Average number of gifts grabbed per hacker.
4. Average waiting time in the sticker queue.
5. Average waiting time in the hoodie queue.
6. Average number of code commits per hacker.
7. Average change length of the commits, including the ones shorter than 20 lines.
8. Average turnaround time (Turnaround time: Total time passed from the sticker queue entrance until leaving the hoodie desk.) To compute, sum all turnaround times and divide it by the number of turnarounds, which is also equal to the number of total gifts given away.

9. Total number of invalid attempts to enter sticker queue.

10. Total number of invalid attempts to get more than 3 gifts.

11. ID of the hacker who spent the most time in the queues and the waiting time of that hacker in seconds. If more than one hacker spent the same amount of time, choose the one with the smallest ID.

12. ID of the hacker who spent the least time in the queues and the waiting time of that hacker in seconds, among the ones who grabbed three stickers and hoodies. If more than one hacker spent the same amount of time, choose the one with the smallest ID. If there is no hacker that grabbed three stickers and hoodies, print -1 for both.

13. Total seconds passed during the hackathon.

Pam also showed the courtesy to share the expected output for the input in Table 2 with explanations. She also showed the progress of queues and desk services in Table 3 where the numbers in the parenthesis in the"desk" columns represent the number of remaining seconds to retrieve the gift.

Output File	Explanation
2	At the 28.5th second the length of the sticker queue is the maximum and it is 2
1	At the seconds 24, 27, and 30 the length of the hoodie queue is the maximum and equals to 1.
2	2, 1, and 3 are the number of gifts (sticker and hoodie as a pocket) retrieved by each competitor respectively and it is divided by 3 (number of hackers) to get the result.
1.567	Hacker 1 waits for 5.5 seconds from "T=18.5" to "T=24". Hacker 2 waits for 2.9 seconds from "T=18.1" to "T=21". Hacker 3 waits in "T=26" and "T=27". The summation is divided by 6 because there 6 different item retrieval process.
1	Hacker 1 waits for 2 seconds between seconds 27 and 29. Hacker 2 waits for a second between seconds 24 and 25. Hacker 3 waits for 3 seconds during "T=30" and "T=33". The summation again is divided by 6.
4.333	The total number of code commit is divided by the number of hackers.
24.615	The total change length of commits is divided by the total number of commits which is 13.
9.567	Hacker 1 enters sticker queue at "T=11", "T=18.5" and leaves at "T=18", "T=33", respectively. Hacker 2 enters at "T=18.1" and leaves at "T=29". Hacker 3 enters at "T=18", "T=26", "T=38" and leaves at seconds "T=25", "T=37", "T=45, respectively.
1	There is one attempt to enter sticker queue with 2 commits by Hacker 1
1	There is 1 attempt to get 4th items by Hacker 3
1 7.5	Hacker 1 waits for 7.5 seconds which is the maximum.
3 4	There is only 1 hacker who grabbed three stickers and hoodies and waits for 4 seconds
46	Hacker 3 tries to enter the sticker queue at 46. seconds which is the last event in the system.

Table 2: Expected Output File and Explanation of each Statistic

Second	Sticker Queue	Sticker Desk	Hoodie Queue	Hoodie Desk
11	-	Hacker 1 (3)	-	-
14	-	-	-	Hacker 1 (4)
18	-	Hacker 3 (3)	-	-
18.1	Hacker 2	Hacker 3 (2.9)	-	-
18.5	Hacker 2, Hacker 1	Hacker 3 (2.5)	-	-
21	Hacker 1	Hacker 2 (3)	-	Hacker 3 (4)
24	-	Hacker 1 (3)	Hacker 2	Hacker 3 (1)
25	-	Hacker 1 (2)	-	Hacker 2 (4)
26	Hacker 3	Hacker 1 (1)	-	Hacker 2 (3)
27	-	Hacker 3 (3)	Hacker 1	Hacker 2 (2)
29	-	Hacker 3 (1)	-	Hacker 1 (4)
30	-	-	Hacker 3	Hacker 1 (3)
33	-	-	-	Hacker 3 (4)
38	-	Hacker 3 (3)	-	-
41	-	-	-	Hacker 3 (4)

Table 3: Step by step progress of the queues and desks. Numbers in the parentheses denote the remaining seconds to leave the desk.

Some tips regarding the project

1. First, ensure to understand the project with all definitions and concepts before implementing. **DO NOT START CODING RIGHT AWAY!**. When you are comfortable with project in every aspect, it is highly recommended to simulate some test cases by hand and think of edge cases. Then, you will be ready to start coding. Otherwise, you might end up like in Figure 2

2. For a more clean and understandable code, try to code object-oriented. Think about the classes and structs that you can use. An object-oriented design will make the code easier to develop and to debug.

Grading

We will automatically grade your submission with multiple test cases besides the ones we already provided. Therefore, **ensure that your submission is runnable with the commands we provided and satisfies all project requirements.** The auto-runnability of the submission will constitute **10/100** of your final grade and passing the test cases will earn your **90/100** more points. If your submission is not auto-runnable, then **you will receive 0 at first and then have to issue an objection**.

Submission Details

You are supposed to use GitHub Classroom for the submission same as the Project 1. No other type of submission will be accepted. Also pay attention the following points:

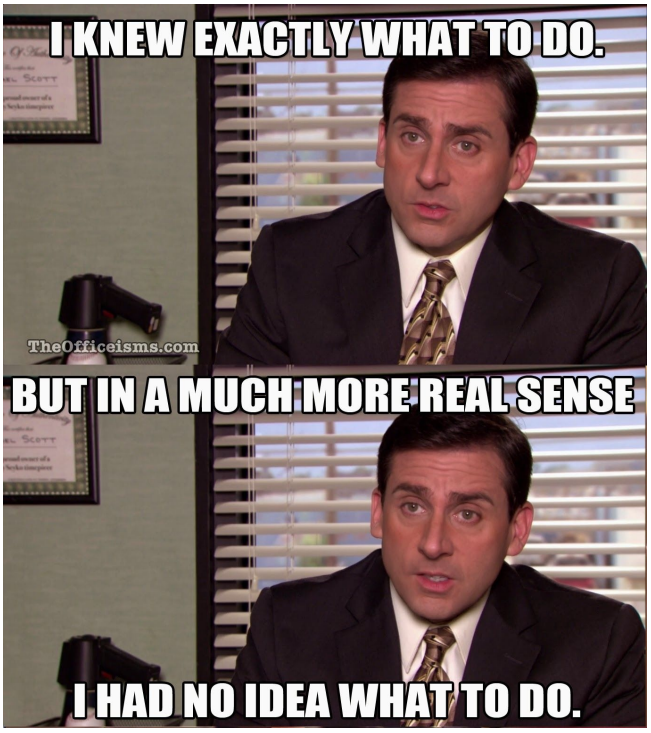

Figure 2: Starting to Code Right Away

- All source codes are checked automatically for similarity with other submissions and exercises from previous years. **Make sure you write and submit your own code**.
- You can add as many files as you can as long as they are in the same folder with main.cpp.
- You are allowed to use **only** the standard C++ library. All requirements of the project must be satisfied with your authentic code, not by any external code. **Otherwise, you will get 0**.
- You are expected to use C++ as powerful, steady, and flexible as possible. Use mechanisms that affect these issues positively.
- Make sure you document your code with necessary inline comments and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long. This is very important for partial grading.
- Try to write as **efficient** (both in terms of space and time) as possible since we will **limit the run times** of the submissions to detect infinite loops and so on.