CS-350 - Fundamentals of Computing Systems
Homework Assignment #5

Due on Oct. 27, 2022 11.59 pm — Late deadline: Oct. 29, 2022 at 11.59 pm at 11:59 pm

Prof. Renato Mancuso

Renato Mancuso

Job ID	Job Length	Arrival time
1	9	0
2	4	2
3	7	1
4	5	4

Table 1: Job parameters.

Problem 1

Consider the mix of jobs with parameters described in Table 1. Assume that ties (if any) are broken by scheduling jobs with lower ID first. Compute the following:

a) The schedule produced by SRT. Try to use the same notation as in the lecture notes.

b) The schedule produced by HSN. Try to use the same notation as in the lecture notes.

c) The schedule produced by SJN. Try to use the same notation as in the lecture notes.

d) The schedule produced by Round-Robin with a quantum of 1 time unit. Try to use the same notation as in the lecture notes.

e) Which algorithm achieves better performance in terms of average response time? Motivate your answer.

f) Which algorithm achieves better fairness? Motivate your answer.

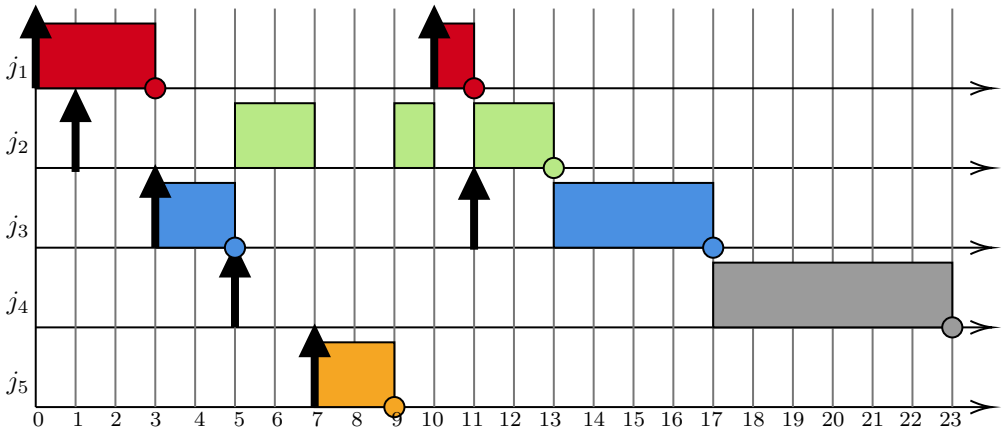

Figure 1: Schedule produced by an unknown scheduler.

Problem 2

Consider the schedule in Figure 1 and answer the following questions. The "dot" in the figure indicates completion of a job, while an up-arrow indicates arrival.

a) What are the parameters (e.g. arrival time, length) of the jobs in this system. Use the task/job notation provided in class and in the notes.

b) What scheduling algorithm is being used to produce the schedule in Figure 1? Motivate your answer.

c) Provide a step-by-step explanation of the decision made by the scheduler at time 11.

d) Use the same job parameters derived above and plot the schedule that would be produced by SJN on the same job parameters.

e) Use the same job parameters derived above and plot the schedule that would be produced by HSN on the same job parameters.

	Website 1		**Website 2**		**Website 3**	
Index	Arrival	Length	Arrival	Length	Arrival	Length
1	0	3	1	4	2	2
2	9	2	9	5	9	3
3	15	4	18	3	16	2
4	28	2	25	2	23	1
5	37	2	30	3	30	3
6	–	–	37	4	37	2

Table 2: List of requests with arrival times and processing time.

Problem 3

A single-cpu web server is trying to be smart about the way it schedules processing for incoming page requests. The web server is running a sharing hosting service with three websites deployed on it. As page requests come in for the three websites, the web server classifies the requests depending on the page they target. It then tries to schedule the pending requests for each of the websites using Shortest Job Next. But the server does not know the future! This means that it needs to estimate, for each class of requests, how long the next request will last based on what it observed in the past.
Table 2 reports the *ground truth* on the actual runtime of a series of about 6 requests per website and their arrival times. Remeber: the webserver scheduler does not know the exact length of a job reported in the table until the job has completed. Also assume that when the scheduler has no knowledge of the requests at all, it will default to schedule a request for website 1 first, then website 2, and so on. This is also the priority ordering in case of any tie. Remeber also that if at any time there is more than one ready request for the same website that is ready, the FIFO ordering is followed.

a) Assume an impossible webserver that is able to foresee the future (the oracle server!), what would be the order in which the various requests are processed? Produce a time plot of the resulting schedule that goes until the last request has been completed.

b) Now consider a realistic scheduler without clairvoyance capabilities. What is the resulting schedule if the server tries to predict the length of the next request directed to each website using an exponentially weighted moving average with parameter $\alpha = 0.5$? Show your work where it is clear at every step of the way how the scheduler it making predictions and taking decisions.

c) Compare now the two schedules produced above. Did not knowing the future caused a performance degradation in terms of average response time?

d) Once again compare the two schedules and focus on the website that statistically receives shorter-lived requests. How was the slowdown of the requests for this website impacted by the scheduler's inability to predict the future?

e) Without repeating the steps above, and focusing only on the difference between predictions and ground truth (prediction error), can we find a better value for α? Analyze the average prediction error for the case considered above, and then for 0.3 and 0.8. Which one of the three cases you expect to be in closer match with what you drew in Part a)

Problem 4

Code: In this problem you will write two functions to work with MD5 hashes. The first function takes a number and computes a MD5 hash. The second function takes a MD5 hash and returns the number that, when hashed, produces the hash in input.

a) Start by writing a Java class namely `Hash.java` that implements number hashing logic. The class should have a method with the following prototype: `String hash(int to_hash)`, produces the MD5 hash string for the integer number provided in input. Internally, the integer should be converted to a string that represents the number in decimal format, and then hashed using the MD5 cryptographic hash.

For instance, the return value of `hash(12345)` should be "827ccb0eea8a706c4c34a16891f84e7b". You are allowed to use Java libraries for the computation of MD5 hashes.

Next, write a Java class namely `UnHash.java` that implements number de-hashing (a.k.a. hash cracking) logic. The class should have a method with the following prototype: `int unhash(String to_unhash)`, produces an integer from a hash string in input. The integer produced in output should be such that its MD5 hash corresponds to the hash string `to_unhash`.

For instance, the return value of `unhash(''01cfcd4f6b8770febfb40cb906715822'')` should be 54321. You are allowed to use Java libraries for the computation of MD5 hashes.

Apart from implementing the `unhash(...)` method, the class should also include a `public static void main(String [] args)` function. The `main(...)` function should accept 1 parameter from the calling environment. The parameter is a string that contains an hash string to crack.

It is responsibility of the `main(...)` function to internally invoke the implemented `unhash(...)` function **only once** and print its result in decimal format.

b) Using the code written in the previous part, work on being able to crack a list of MD5 hashes from a file. For this part, write a Java class called `Dispatcher.java`. The input data (list of hashes to crack) is given in a file. The path to the file is passed as the only parameter by the calling environment. The input file is structured as a list of MD5 hashes to crack, one per line, with each line terminated with a newline (\n) character.

It is the job of the dispatcher to (1) read the input file; (2) invoke the `unhash(...)` procedure written for the first part for each of the hashes in the input file. The result of each of the unhash operations, i.e. the cracked hashes, should be printed in output, with a single line per decoded hash. Apart from the list of decoded hashes, nothing else should be printed in output.

Submission Instructions: in order to submit this homework, please follow the instructions below for exercises and code.
The solutions for Problem 1-3 should be provided in PDF format, placed inside a single PDF file named `hw5`.pdf and submitted via Gradescope. Follow the instructions on the class syllabus if you have not received an invitation to join the Gradescope page for this class. You can perform a partial submission before the deadline, and a second late submission before the late submission deadline.

The solution for Problem 4 should be provided in the form of Java source code. To submit your code, place all the `.java` files inside a compressed folder named `hw5`.zip. Make sure they compile and run correctly according to the provided instructions. The first round of grading will be done by running your code. Use CodeBuddy to submit the entire `hw5`.zip archive at `https://cs-people.bu.edu/rmancuso/courses/cs350-fa22/codebuddy.php?hw=hw5`. You can submit your homework multiple times until the deadline. Only your most recently updated version will be graded. You will be given instructions on Piazza on how to interpret the feedback on the correctness of your code before the deadline.