CS-350 - Fundamentals of Computing Systems
Homework Assignment #6

Due on November 7, 2020 at 11:59 pm

Prof. Renato Mancuso

Renato Mancuso

	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 1: List of requests with arrival times and processing time.

Problem 1

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 **??** reports the *ground truth* on the actual runtime of a series of 5 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.

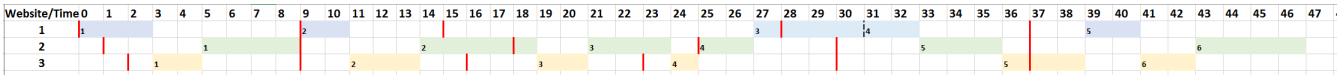


Figure 1: Ans-1a: Shortest Job First

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.

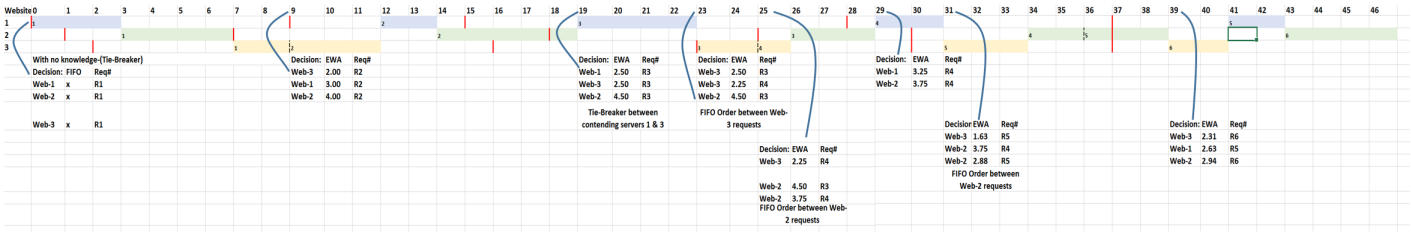


Figure 2: Ans-1b: Shortest Job First with predictions based on EWA

	Web-1			Web-2			Web-3		
Index	Arrival	Ground Truth	EWA	Arrival	Ground Truth	EWA	Arrival	Ground Truth	EWA
1	0	3	3.00	1	4	4.00	2	2	2.00
2	9	2	3.00	9	5	4.00	9	3	2.00
3	15	4	2.50	18	3	4.50	16	2	2.50
4	28	2	3.25	25	2	3.75	23	1	2.25
5	37	2	2.63	30	3	2.88	30	3	1.63
6	x	x	x	37	4	2.94	37	2	2.31
average		**2.60**	**2.88**		**3.50**	**3.68**		**2.17**	**2.11**
alpha	0.5								

Figure 3: Ans-1b: EWA for the three websites

Ans-1b The calculations of predicted times using EWA method are provided in Figure 3. Based on these we construct a SJN schedule in Figure 2. The decisions based on the predicted times are indicated accordingly on the graph at the time instances when the scheduler is invoked (indicated with blue curves).

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

Absolute	Web-1	Web-2	Web-3	**Predicted**	Web-1	Web-2	Web-3
Index	Response Time			Index	Response Time		
1	3	8	3	1	3	6	7
2	2	10	5	2	5	10	3
3	16	6	5	3	8	11	9
4	5	2	2	4	3	11	3
5	4	6	9	5	6	9	4
6	x	10	6	6	x	10	4
sum	30	32	24	sum	25	47	26
overall avg response time per req	**5.06**			**overall avg response time per req**	**5.76**		

Figure 4: Ans-1c: Response Time Comparison of the two schedules from part 1a and 1b

Ans-1c Referring to Figure 4, yes a degradation results. We see an increase in the overall average response time per request of the predicted schedule as opposed to that based on the ground truth.

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?

Ans-1d Website 3 receives statistically shorter-lived requests as can be seen by the average processing times (Figure 3). We calculate slowdown for each request targeting Website-3 for the two schedules. The schedule using the EWA strategy to predict the processing times results in greater slowdown per request on average as opposed to the oracle schedule.

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)

Ans-1e Refer to Figure 6 for calculations of prediction error for all 3 values of α. Given that $\alpha = 0.8$ yields the smallest average prediction error per request, it follows that it would yield a schedule that matches more closely with the Oracle schedule of Part a).

Web-3 Req Index	Ground Truth Requested Time	Oracle Response Time	slowdown	Predicted Requested Time	EWA Response Time	slowdown
1	2	3	1.50	2.00	7	3.50
2	3	5	1.67	2.00	3	1.50
3	2	5	2.50	2.50	9	3.60
4	1	2	2.00	2.25	3	1.33
5	3	9	3.00	1.63	4	2.46
6	2	6	3.00	2.31	4	1.73
average			**2.28**			**2.35**

Figure 5: Ans-1d: Slowdown Comparison of the two schedules for Website-3

Web-1							**Web-1**		
			EWA				**Prediction Error**		
Index	Arrival	Ground Truth	alpha=0.5	alpha=0.3	alpha=0.8		alpha=0.5	alpha=0.3	alpha=0.8
1	0	3	3.00	3.00	3.00		0.00	0.00	0.00
2	9	2	3.00	3.00	3.00		1.00	1.00	1.00
3	15	4	2.50	2.70	2.20		-1.50	-1.30	-1.80
4	28	2	3.25	3.09	3.64		1.25	1.09	1.64
5	37	2	2.63	2.76	2.33		0.63	0.76	0.33
6	x	x	x	x	x		x	x	x
Web-2							**Web-2**		
			EWA				**Prediction Error**		
Index	Arrival	Ground Truth	alpha=0.5	alpha=0.3	alpha=0.8		alpha=0.5	alpha=0.3	alpha=0.8
1	1	4	4.00	4.00	4.00		0.00	0.00	0.00
2	9	5	4.00	4.00	4.00		-1.00	-1.00	-1.00
3	18	3	4.50	4.30	4.80		1.50	1.30	1.80
4	25	2	3.75	3.91	3.36		1.75	1.91	1.36
5	30	3	2.88	3.34	2.27		-0.13	0.34	-0.73
6	37	4	2.94	3.24	2.85		-1.06	-0.76	-1.15
Web-3							**Web-3**		
			EWA				**Prediction Error**		
Index	Arrival	Ground Truth	alpha=0.5	alpha=0.3	alpha=0.8		alpha=0.5	alpha=0.3	alpha=0.8
1	2	2	2.00	2.00	2.00		0.00	0.00	0.00
2	9	3	2.00	2.00	2.00		-1.00	-1.00	-1.00
3	16	2	2.50	2.30	2.80		0.50	0.30	0.80
4	23	1	2.25	2.21	2.16		1.25	1.21	1.16
5	30	3	1.63	1.85	1.23		-1.38	-1.15	-1.77
6	37	2	2.31	2.19	2.65		0.31	0.19	0.65
						average	**0.13**	**0.17**	**0.08**

Figure 6: Ans-1e: Prediction Error= Prediction Time - Ground Truth

Problem 2

You are doing your work-from-home internship at Eastern Digital (ED) and were recently tasked with analyzing the performance of one of their latest-and-greatest hard disk drives (HDDs). But because ED oursourced the implementation of the disk scheduler offshore, nobody at ED knows exactly what block scheduling algorithm is internally implemented by the HDD. You are here to save the day and reverse-engineer the disk.

You decided to send a set of requests to the HDD and monitor the movement of the disk head to better understand the behavior of the scheduler. You know from the specs that the HDD has 12 tracks numbered from 0 to 11. You also measured that it takes the disk head 1 unit of time to move from one track to the next. The requests that you experimented with arrive at different time instants and target different disk tracks, following what reported in Table 2. The disk head at $t = 0$ is at track 2.

You have tracked the movement of the disk-head as it serves the requests in the table, and have plotted this movement as shown in Figure 7. The black arrows represent the request arrivals, the white circles indicate the target track, and the black circles denote the instant when a request is served by the disk head.

Based on all the information provided and all the observations you have made, answer the following questions.

NOTE: always assume that once the disk is in motion to serve a request, a new decision cannot be taken

Request	Arrival time	Track ID
R_1	0	4
R_2	2	6
R_3	4	5
R_4	4	3
R_5	7	10
R_6	11	10
R_7	12	9
R_8	12	7
R_9	18	1
R_{10}	20	2
R_{11}	25	8
R_{12}	29	7

Table 2: Disk request parameters

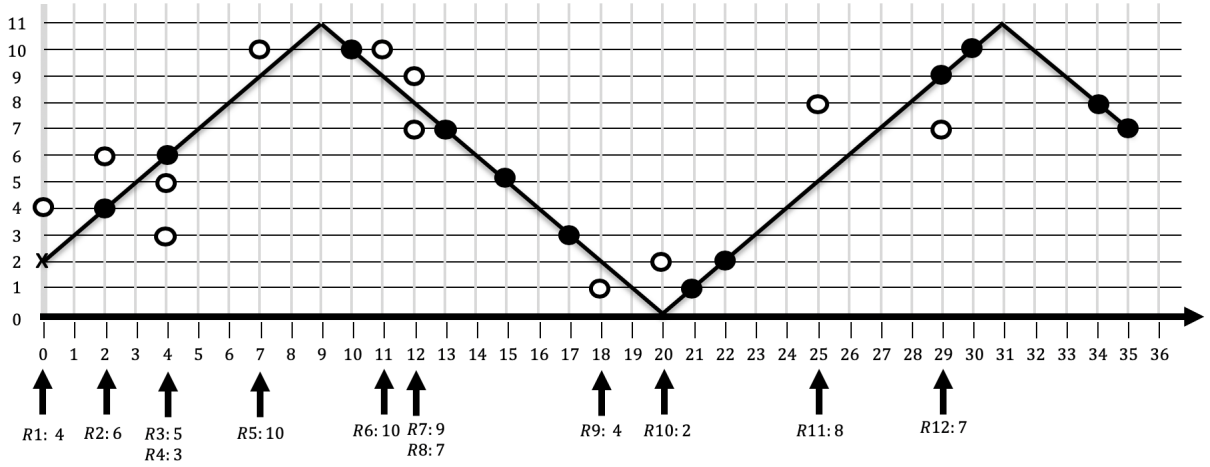


Figure 7: Disk head movement resulting from the sequence of requests in Table 2

until the scheduled movement is complete.

a) Describe the scheduler embedded within the HDD by mentioning (i) the scheduler invocation policy and (ii) the scheduling policy including the condition for changing the head direction.

Ans-2a

(a) There are 2 ready queues. A normal queue and a late request queue.

(b) A request is moved to the late request queue if it has not been served for 11 time units or more

(c) The disk head moves always up→down→up continuously scanning all the tracks

(d) If the late request queue is empty, requests on the way int he normal queue are picked up and served

(e) If the late request queue is not empty, requests in the normal queue are ignored until the late request queue is emptied. Until then the disk only picks up late requests on the way.

(f) In any case it the target cylinder of a pending request is hit twice within 3 time units, the request will be served at the latest possible time.

Note to Graders: For (i) any answer is acceptable as long as the student gives adequate reasoning referring to the schedule given in the Figure

b) While keeping the same exact overall scheduling logic, you notice that certain disk movements are unnecessary. If we get rid of those, can we reduce the response time of the scheduler? And by how much.

Ans-2b Same scheduling logic keeps the order of requests served the same i.e from Figure 7 the order is as follows: $R1 - R2 - R5 - R8 - R3 - R4 - R9 - R10 - R7 - R6 - R11 - R12$. Removing unnecessary disk movements gives us the schedule depicted in Figure -8. The average response time of the scheduler becomes: 7. The original response time of the given schedule in Figure 7 was: 7.33 resulting in a reduction by 4.5%.

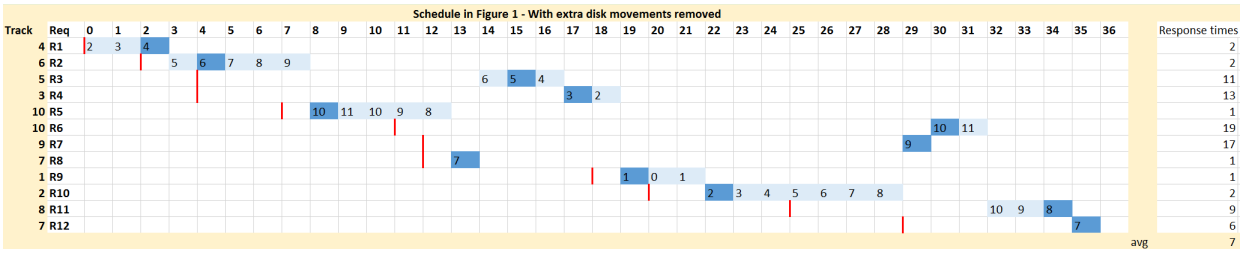


Figure 8: Removing Unnecessary Disk head movement from the schedule in Figure 7

c) Great! After having pointed out that the current scheduler is not great, your boss has tasked you with the job of figuring out a better disk scheduler for the HDD to roll out in the next firmware update. Analyze (a) a Shortest-Scan Next strategy and (b) a First-Ready, First-Come-First-Served strategy where a request is considered "ready" if the position of the disk head at the time of scheduling is within ± 3 tracks. Consider (i) average response time and (2) total disk-head travel length.

Ans-2c Refer to Figure 9 for schedules following a) SSF and b) FR-FCFS strategies. Both strategies yield the same schedule as we assume that the scheduler is invoked at the end of a scheduled movement. Based on the ready requests in the queue at the time of the scheduler invocation, a movement decision is taken and the scheduler is not invoked until the previous movement is complete. Note that FR-FCFS prioritizes requests that are ready i.e. within ± 3 tracks over other requests, including older ones. If no request is ready, then FR-FCFS prioritizes older requests over younger ones. This results in the same schedule as SSF. The average response time is 3.83 and the total disk head movement time is 32.

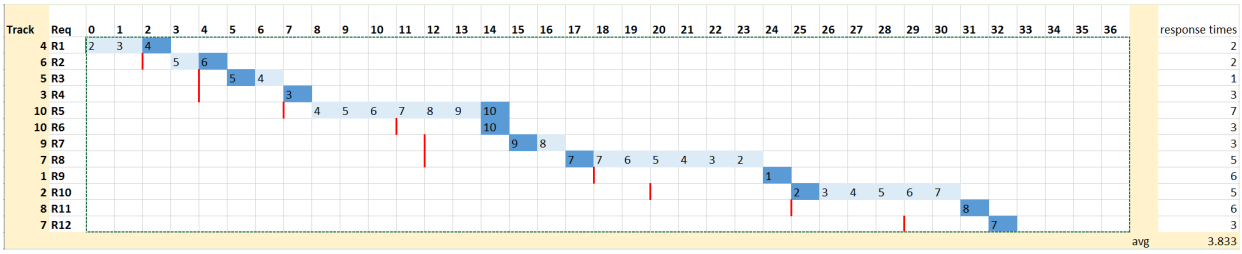


Figure 9: Problem 2c: SSF and FR-FCFS schedule

Problem 3

The F-35 is the most advanced fighter aircraft built to date. The F-35 is built to be physically unstable (much like an inverted pendulum) and it requires software to be continually stabilized. This property makes the aircraft extremely agile, but it also makes the on-board avionics extremely critical to maintain the aircraft in flight. On top of that, a number of advanced software subsystems have been designed to assist the pilot(s) carrying out their mission. We hereby provide a description of the real-time workload deployed on the F-35. All the WCETs are expressed in millions of clock cycles (MCC) instead of absolute time units.
First, a Sensor Fusion (SF) task is responsible for all the data acquisition and aggregation from the various sensors in the aircraft. It runs every 2.5 ms and its WCET is 0.8 MCC. Next, the Assisted Stabilization (AS) is responsible for maintaining the attitude of the aircraft as imposed by the control inputs provided by the pilot. The AS runs for 2.3 MCC every 7 ms. The Active Electronically Scanned Array (AESA) is an advanced radar to assist pilots in engaging pilots in long-range air and ground targets. An AESA task is 5 MCC long and scans are performed at 5 Hz. Next, the Distributed Aperture System (DAS) is a complex subsystem for 360-degrees threats detection system, structured as multiple tasks: (1) missile detection and tracking (MDT) running at 100 Hz, each task with WCET 4 MCC; (2) launch point detection (LPD) with WCET 6 MCC, operating at 50 Hz; (3) fire control capability monitor, running at 80 Hz with worst-case runtime of 3 MCC; and (4) friendly vehicle (FVT) tracking performed every 25 ms, each tracking instance taking at most 5 MCC. Finally, the Communication, Navigation, and Identification (CNI) subsystem handles various voice and data communication functions. This is comprised of a real-time task that runs every 20 ms for a WCET of 12 MCC.

a) Summarize the real-time tasks in an F-35 aircraft with a table. For each task, report its value of C_i, T_i. The table should list the various tasks in the exact order in which they are mentioned in the descriptive paragraph above.

Task	C (WCET) /MCC	C / s	T (time period) /ms	$u_i(\times\frac{10^9}{f})$	u_i (f=1GHz)
SF	0.8	0.8M/f	2.5	0.32	0.32
AS	2.3	2.3M/f	7	0.33	0.33
AESA	5	5M/f	200	0.025	0.025
MDT	4	4M/f	10	0.4	0.4
LPD	6	6M/f	20	0.3	0.3
monitor	3	3M/f	12.5	0.24	0.24
FVT	5	5M/f	25	0.2	0.2
CNI	12	12M/f	20	0.6	0.6

Table 3: List of tasks in F35. Processor clock frequency assumed to be f Hz.

b) Imagine being the system engineer responsible for the integration of all the real-time workload on the F-35, and that you can pick a processor with arbitrary clock speed. Keep in mind that a processor with 1 MHz clock speed is able to crunch 1 MCC per second. How fast you need to pick your processor if you will be using EDF? What if you will be using RM?

Problem 3 continued on next page...

Ans3b With EDF:

$$\begin{aligned} \sum_{i=1}^{8} \frac{C_i}{T_i} &\leq 1 \\ \frac{10^9}{f} \times 2.415 &\leq 1 \\ \implies f &\geq 2.415\ GHz \end{aligned} \tag{1}$$

With RMS(n=8):

$$\begin{aligned} \sum_{i=1}^{8} \frac{C_i}{T_i} &\leq 8(2^{1/8} - 1) = 0.724 \\ \frac{10^9}{f} \times 2.415 &\leq 0.724 \\ \implies f &\geq 3.34\ GHz \end{aligned} \tag{2}$$

Note to Grader: If students use ln 2≈0.69 as the RMS utilization bound then take some marks off their answer.

c) Assume EDF scheduling. The cost of a CPU operating at 1 GHz is \$50 = C. A processor with a speedup of X over that base frequency costs \$$C^X$. Would you go for a single-processors solution or for a multi-processor solution, where each processor operates at 1 GHz?

Ans-3c With EDF from part(b) we note that schedulability is ensured for a uni-processor when clock frequency is at-least 2.415 GHz. This implies a 1GHz processor needs a speedup of x=2.415. Thus total cost with a uniprocessor solution = $50^{2.415} = 12,676.9$ \$. On the other hand if multiprocessors are used each with a clock frequency of 1 GHz then we need at-least 3 processors. Thus cost for such a solution is $50 \times 3 = 150$ \$. Since multiprocessor solution is less pricey, we go for multi-processors.

d) Assume EDF scheduling. Imagine you decide to go for a multi-processor solution. First, pick the minimum number of processors required to consolidate the real-time workload of the aircraft. Next, produce a task-to-processor assignment that ensures that all the tasks will be schedulable.

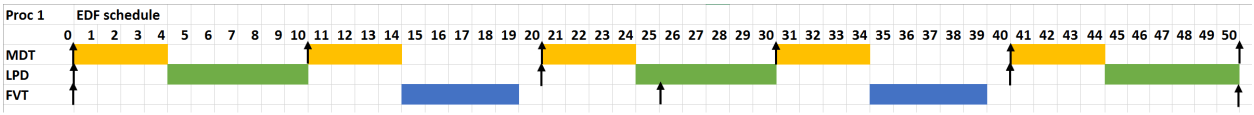
Ans-3d Number of processors required: 3 where each processor operates at 1GHz clock frequency. Using f=1GHz in Table: 3, we compute utilization of each task and report it in a new column in the same table. Using a first fit assignment of tasks to processors, one possible assignment is as follows. Utilization of each processor is reported in the last row of Table: 4.

Proc 1	Proc 2	Proc 3
SF	MDT	CNI
AS	monitor	
AESA	FVT	
LPD		
0.975	0.84	0.6

Table 4: Distribution of Tasks in a 3-processor system(f=1GHz) under static mapping based on EDF scheduling

Note to graders: As long as each processor utilization ≤ 1, any assignment is acceptable.

e) Once again consider EDF scheduling, and assume that the first 1 GHz processor has been assigned tasks MDT, LPD, and FVT. Produce the schedule generated from time 0 and up to time 50.

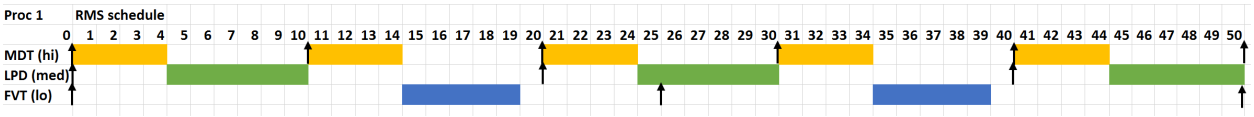


f) Consider now RM scheduling, with the same tasks assigned to the first 1 GHz processor (MDT, LPD, and FVT). Is the system schedulable? Motivate your answer.

Ans-3f

$$\sum_{i=1}^{3} \frac{C_i}{T_i} = 0.4 + 0.3 + 0.2 = 0.9. \tag{3}$$

Since the RMS utilization bound for the three tasks is $3(2^{1/3} - 1) = 0.78$ and $0.9 > 0.78$, we cannot make any conclusive remarks about RMS schedulability. However with RMS based static priority assignment: MDT (p=high), LPD (p=medium) and FVT (p=low), the following schedule shows no deadlines are missed thus the task set is still schedulable under RMS (for the first 50ms at-least).



Problem 4

Code: In this problem you will write two functions to navigate complex structures that are based on the ability to crack MD5 hashes.

a) Extend the class `Dispatcher.java` that you implemented as part of HW5 to parallelize the cracking of a set of hashes. Use Java support for multi-threading to define as many woorker threads as the number of CPUs N in the machine—passed as a parameter to your code. The Dispatcher should construct a global work queue where each hash to crack is a unit of work and provided to the first available worker for cracking. Define a `dispatch(...)` method to handle the distribution of work units to worker threads.

Apart from implementing the `dispatch(...)` method, the class should also include a `public static void main(String [] args)` function. The `main(...)` function should accept 2 parameters from the calling environment. First, the input data (list of hashes to crack) is given in a file. The path to the file is passed as the first 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. Second the number of CPUs available on the machine N is passed as the second parameter.

It is responsibility of the `main(...)` function to internally initialize and start the dispatcher. It is the job of the dispatcher to (1) read the input file; (2) distribute the work to the worker threads and let the threads print the result. 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.

b) Modify the dispatcher to be able to handle impossible hashes. For this part, you will be given an input file that contains a few hashes that are impossible to crack. Extend the dispatcher to handle timeouts in the processing of hashes to that the workers do not get stuck trying to reverse impossible hashes.

For this part, the input file will appear identical to the previous part, at least on the surface. The second parameter passed to your code will still be the number of available CPUs. A third parameter is added for this part, that encodes the length of the timeout for impossible hashes expressed in milliseconds.

Just like the previous part, it is responsibility of the `main(...)` initialize and start the dispatcher. It is the job of the dispatcher to (1) read the input file; (2) distribute the work to the worker threads and let the threads print the result. Importantly, (3) the dispatcher should handle the timeout for all the threads. If a worker does not complete on time, the dispatcher should order the thread to move on. In this case, the uncrackable hash as it was read from the input file should be print in output.

HINT: in Java, thread signaling works in a very counter-intuitive way. Use variables to control the execution flow on the worker threads instead.

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 `hw6`.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 `hw6`.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 `hw6`.zip archive at `https://cs-people.bu.edu/rmancuso/courses/`

`cs350-fa20/scores.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.