Computer Science Department

CS-350: Fundamentals of Computing Systems

Final

Fall 2019

Name: ______________________________

Login: ____________________ BU Id: ____________________

You can use the back of any page to answer the question on the front of that page

Problem #1:	*/26*
Problem #2:	*/20*
Problem #3:	*/20*
Problem #4:	*/20*
Problem #5:	*/20*
Problem #6:	*/14*
Total Grade:	*/100*

Note:

- This is a closed-book/notes exam
- Basic calculators are allowed
- Time allowed is 120 minutes
- There are 120 total points
- Problems are weighted as shown
- Sub-problems are weighted as reported
- Explain your assumptions clearly

I give credit to students who can accurately assess their performance. Thus, after you finish the exam answer the following for up to five bonus points!

What do you guess your score will be? __________

Bonus = max(0 , 5 - | *GuessedGrade – ActualGrade* |)

1) Label each of the statements below with either **True** or **False**:

	Question	**Answer**
a.	Map-reduce is a computational paradigm for achieving massive speedups when processing workloads the exhibit good parallelism.	
b.	Consider a set of real-time tasks with 85% utilization scheduled with EDF and that use semaphore-based synchronization. The taskset is definitely schedulable.	
c.	In a distributed system using Lamport clocks for synchronization, it is always possible to establish an “happened after” relationship for any two events.	
d.	Consider an M/M/1/K system where K approaches infinity. It is okay to approximate the system with an M/M/1 model.	
e.	In a distribute system, one can achieve synchronization among processes running on multiple machines using traditional semaphores.	
f.	Shortest Job Next is a good scheduling algorithm when the length of the jobs is known because it minimizes response time and it is free from starvation.	
g.	CFQ scheduling disk operations requires knowledge of the exact disk geometry (i.e., structure of cylinders, clusters, etc.) to be correctly implemented.	
h.	Consider a real-time taskset scheduled using EDF. If a deadline miss is detected, it must be true that X time units of CPU time were requested over an interval of time of length Y, for some value of X and Y with X > Y.	
i.	It is possible for the same stateful resource subject to the same load to exhibit different utilizations when managed with two different scheduling algorithms.	
g.	When computing an Exponentially Weighted Moving Average, selecting a value of α closer to 1 results in an average that quickly forgets past values.	
k.	In an Out-Of-Order multi-core processor one can use the Peterson’s algorithm to achieve 2-party mutual exclusion, but not the Dekker’s algorithm.	
l.	Consider a single-CPU system with 1 running process P and no ready processes. P uses a semaphore *sem* that was initialized to 0 and with current value -3. After P performs a signal(*sem*) operation, there will be at least one ready process.	
m.	Deadlocks cannot occur in a single-processor fully non-preemptive system, i.e. where, once started, a process is always run to completion with no interruption.	
n.	Deadlocks cannot occur in a multi-processor fully non-preemptive system, i.e. where, once started, a process is always run to completion with no interruption.	
o.	A spinlock differs from a semaphore in that it only allows mutual exclusion and because a process waiting to acquire a spinlock will keep the CPU busy.	
p.	Implementing an N*M/M/1 system can be done without the need for N-party synchronization. The same is not true for an M/M/N system.	
q.	A system uses the Banker algorithm. If a resource allocation request is denied, it means that having granted it would have definitely resulted in a deadlock.	
r.	Take a system with an infinite queue, Poisson arrivals, single processor, and service time that is uniformly distributed between A and B (known). The system can be only simulated but not studied analytically.	

Note: There are 18 questions. A correct answer will get you +2 points. An incorrect answer will get you -1 points. A blank answer will get you 0 points.

2) (20 points) A function-as-a-service (FaaS) computing cluster is comprised of 4 single-processor machines. Requests from users are functions that arrive at a first user-facing server called the "Dispatcher" (D). It analyzes the resources required by the request, which takes on average 12 msec. If only processing is required, which happens with probability 0.1, the request is handed to the processing engine (P) where it spends on average 130 msec. With 0.6 probability, the dispatcher might determine that an in-memory object needs to be retrieved before the request can proceed. In this case, the request goes to the "Memory" (M) server which has an average service time of 18 msec. In all the other cases, after the dispatcher, a file needs to be fetched from storage to handle the request. In this case, it will be forwarded to the "storage" (S) server which has an average service time of 38 msec.
After a request completes at P, it always exits the system. After a request leaves the M server, if no further processing is required, the request can exit the system right away. If more processing is required, it will go back to the dispatcher as if it was a new request. The latter (i.e., going back in) happens with probability 0.2. Lastly after a request leaves S, it will always go back to the dispatcher as if it was a new request. On average the system is receiving 40 new requests every second.

a) **(5 points)** Draw a diagram of the system as a network of queues. Specify the type of model you are using for your servers and the assumptions required for the model to be valid.

b) **(5 points)** Compute the utilization for the four servers.

c) **(5 points)** Compute the overall average response time of a generic request.

d) **(5 points)** You have the option of upgrading one of the servers with a 2-processor machine, where each processor would have a private queue of requests. Determine which one of the four servers should be upgraded and describe **how that would improve the maximum sustainable request rate** of the system (i.e. the maximum rate of requests the system can handle without "exploding").

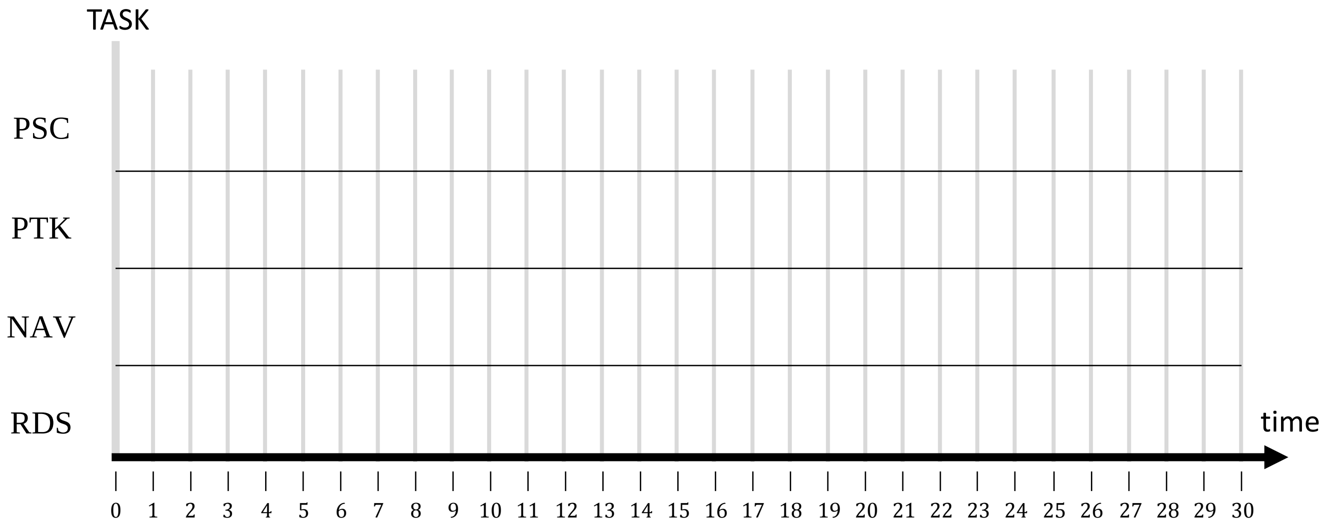
3) (20 points) Elon Tusk is about to roll out the new Hypertruck that features a full-fledged autopilot. All the individual components of the system, each implemented as a periodic task, have been developed and now need to be integrated. These are (1) the pedestrian trajectory tracking (PTK) system; (2) the road sign detection subsystem (RDS); (3) the propulsion and steering control task (PSC); and (4) the navigation and trajectory planning system (NAV). To cut costs, the Hypertruck features a single-processor on-board computer. The parameters of the 4 tasks are summarized in Table 1.

Task Name	WCET	Period	Utilization
PSC	1 msec	5 msec	0.2
PTK	3 msec	10 msec	0.3
NAV	3.5 msec	15 msec	0.23
RDS	4 msec	20 msec	0.2

Table 1: Task parameters of ICU system.

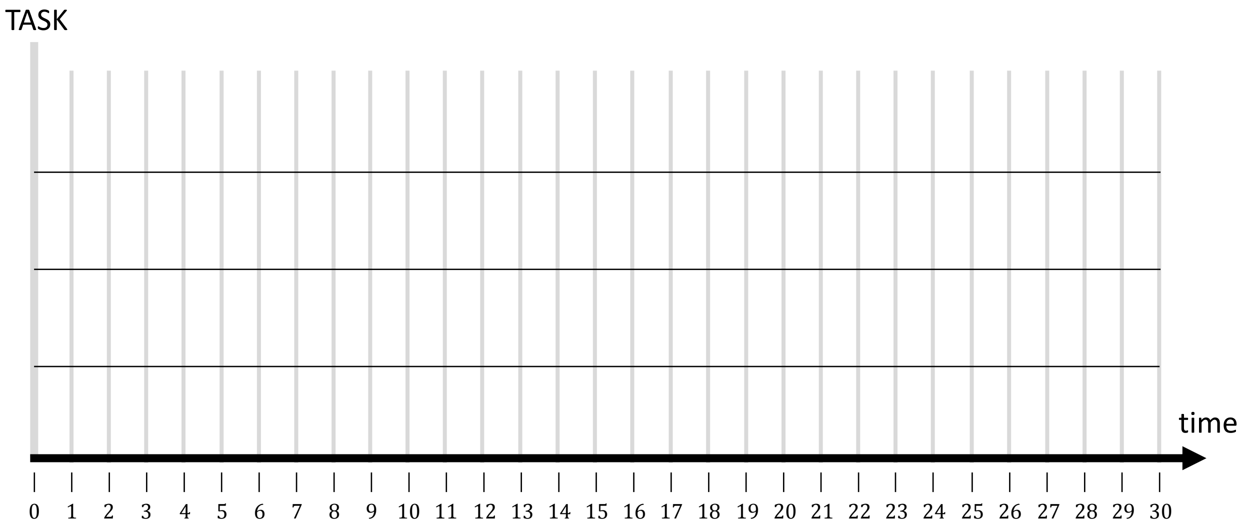
a) **(2 points)** Complete the table above with the utilization of the four tasks.

b) **(6 points)** Draw the schedule that RM would produce from time 0 until 30. Assume that all the tasks initially arrive at time 0.



c) (**6 points**) The engineers working on the Hypertruck would like to use RM to schedule the tasks on the onboard computer. Is it safe to do so? Explain why to the engineers. If not, propose a solution which only involves changing the scheduling strategy.

d) **(6 points)** After hearing your argument, one of the engineers came back with a counter-proposal. He is suggesting collapsing two consecutive jobs of task NAV into one with a period that is twice as long. As such, the new WCET of a NAV job would be 7 msec, with a period of 30 msec. Is this a good solution? Motivate your answer. NOTE: *you may use the grid provided below if needed.*



4) (20 points) You are sitting in the control room of the MBTA Green Line and you are specifically been assigned the task of measuring the quality of service for the segment Kenmore Square (K) to Hynes Convention Center (H). You have measured 1024 times the time it takes for trains travelling from K to H. The average trip time was measured to be 135 *sec*, with a measured **variance** of $S^2 = 12100\ sec^2$.

a) (6 points) What is the 99% confidence interval on the time it takes for a train ride from K to H?

b) (7 points) With what confidence you can inform passengers that the train will take somewhere between 130 and 140 seconds to bring them from K to H?

c) (**7 points**) How many more rides you should measure to bring the confidence on the [130, 140] ride time interval to 99%?

5) (**20 points**) A CS350 student has worked hard to parallelize their code to be the champion of the Great Pirate Challenge. To do so, their code defines two shared queues, one for new hashes to be cracked, and one for cracked hashes; and a tree data structure. The queues and the tree are synchronized using three semaphores. The code defines three type of threads: a Main Thread, a Coordination Thread, and a UnHash Thread. The fragment of code handling synchronization for the three threads is provided below.

```
Initialization:

   Semaphore iqueue = 1;
   Semaphore oqueue = 1;
   Semaphore tree   = 1;
```

```
Process Main Thread:
DO:
   wait(tree);
   wait(iqueue);
   /* Get new offsets from tree */
   /* Add new hashes to input queue*/
   signal(iqueue);
   signal(tree);
FOREVER
```

```
Process UnHash Thread:
DO:
   wait(iqueue);
   wait(oqueue);
   /* Get new input hash */
   /* Crack input hash */
   /* Add result to output queue */
   signal(oqueue);
   signal(iqueue);
FOREVER
```

```
Process Coordination Thread:
DO:
   wait(oqueue);
   wait(tree);
   /* Take results from out queue */
   /* Update tree with new offsets */
   signal(tree);
   signal(oqueue);
FOREVER
```

a) **(8 points)** You noticed that in some cases your code never completes, while on the same input other times it just works as expected even when executed on a single processor. Assume a single processor and use the table below to show an interleaving of operations performed by the threads that leads to the problem. **NOTE**: *only one operation per row should be filled. The first row is provided for your convenience. Use only the rows you need.*

step	Main Thread	UnHash Thread	Coordination Thread
1	wait(tree);	//////////	//////////
2			
3			
4			
5			
6			
7			

b) **(6 points)** Devise a solution revolving around the fact that only 2 out of three threads can be attempting at acquiring the shared resources they need at the same time. For this purpose, add a new semaphore called MPL. Complete the new initialization block and show how the new semaphore would be used in the code of the Main Thread (assuming that the same will be done in the other threads). **NOTE**: *when completing the Main Thread's code, only fill the blank spaces you deem necessary. Not all the blank spaces need to be filled.*

```
Initialization:

   Semaphore iqueue = 1;
   Semaphore oqueue = 1;
   Semaphore tree   = 1;

   Semaphore MPL = ____________;
```

```
Process Main Thread:
DO:

   __________________________;
   wait(iqueue);

   __________________________;
   wait(oqueue);

   __________________________;
   /* Get new input hash */
   /* Crack input hash */
   /* Add result to output queue */

   __________________________;
   signal(oqueue);

   __________________________;
   signal(iqueue);

   __________________________;
FOREVER
```

c) **(6 points)** Devise a solution where no new semaphore is introduced, and where only the order in which resources are acquired by the threads is rearranged. Complete the fragment of code below for the three threads. Only specify which semaphore is acquired in each wait(…) operation. Only change the order of acquisition of the semaphores, not which semaphores are used by each of the threads.

```
Process Main Thread:
DO:

   wait(______________);

   wait(______________);

   ...
```

```
Process UnHash Thread:
DO:

   wait(______________);

   wait(______________);

   ...
```

```
Process Coord. Thread:
DO:

   wait(______________);

   wait(______________);

   ...
```

6) (**14 points**) Consider a system with 4 Processes $P_1, P_2, P_3,$ and P_4 and 3 shared resources $R_1, R_2,$ and R_3. For these resources, the static parameters are reported in Table 3.

		Parameter	Resources		
			R_1	R_2	R_3
		$R(k)$	6	11	7
Processes	P_1	$C_1(k)$	6	3	4
	P_2	$C_2(k)$	3	6	2
	P_3	$C_3(k)$	7	7	3
	P_4	$C_4(k)$	4	7	3

Table 3: Static parameters of considered system.

a) **(7 points)** Considering what reported in Table 3, complete Table 4 with the amount of allocated resources to all the processes $A_i(k)$, and with the availability of all the resources $V(k)$.

		Parameter	Resources			Parameter	Resources		
			R_1	R_2	R_3		R_1	R_2	R_3
		$V(k)$							
Processes	P_1	$A_1(k)$				$N_1(k)$	4	2	2
	P_2	$A_2(k)$				$N_2(k)$	0	3	0
	P_3	$A_3(k)$				$N_3(k)$	7	5	1
	P_4	$A_4(k)$				$N_4(k)$	3	5	3

Table 1: System state for considered system.

b) **(7 points)** Would the state in Table 4 be deemed safe by the Banker algorithm? If so, produce a possible sequence in which all the processes can come to completion.

Some Probability / Statistics Relations:

- $Cor(X,Y) = \dfrac{\frac{1}{N}\sum_{i=1}^{N}(X(i)-\overline{X})(Y(i)-\overline{Y})}{StdDev(X) * StdDev(Y)}$
- Poisson $f(x) = \dfrac{\lambda^x}{x!}e^{-\lambda}$
- Exponential $f(x) = \lambda e^{-\lambda x}$, $F(x) = 1 - e^{-\lambda x}$
- Standard Normal $f(z) = \dfrac{1}{\sqrt{2\pi}}e^{-0.5z^2}$

Equations for some queuing systems:

- M/G/1 system $q = \dfrac{\rho^2 A}{1-\rho} + \rho$ and $w = \dfrac{\rho^2 A}{1-\rho}$, where $A = \dfrac{1}{2}\left[1 + \left(\dfrac{\sigma_{Ts}}{Ts}\right)^2\right]$
- M/D/1 system $q = \dfrac{\rho^2}{2(1-\rho)} + \rho$ and $w = \dfrac{\rho^2}{2(1-\rho)}$
- M/M/1/K system $q = \begin{cases} \dfrac{\rho}{(1-\rho)} - \dfrac{(K+1)\rho^{K+1}}{(1-\rho^{K+1})} & \text{for } \rho \neq 1 \\ \dfrac{K}{2} & \text{for } \rho = 1 \end{cases}$, and

$$\Pr(\text{"Rejection"}) = \Pr(S_K) = \begin{cases} \dfrac{(1-\rho)\rho^K}{(1-\rho^{K+1})} & \text{for } \rho \neq 1 \\ \dfrac{1}{K+1} & \text{for } \rho = 1 \end{cases}$$

- M/M/N system $q = C\dfrac{\rho}{1-\rho} + N\rho$ and $w = C\dfrac{\rho}{1-\rho}$,

$$\text{where } \rho = \frac{\lambda}{N\mu} = \frac{\lambda T_s}{N},\ C = \frac{1-K}{1-\rho K} \text{ and } K = \frac{\sum_{i=0}^{N-1}\frac{(N\rho)^i}{i!}}{\sum_{i=0}^{N}\frac{(N\rho)^i}{i!}}$$

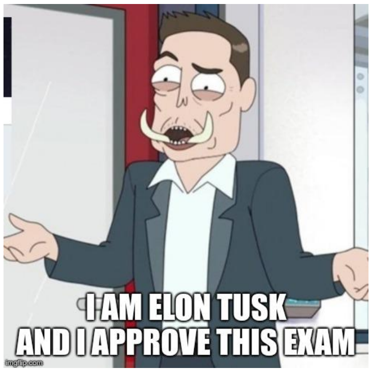
Tasks and Schedulability Results:

- Minimum Slowdown for job $J_i$at time t: $\frac{t-a_i+C_i}{C_i}$
- Single-processor RM: n tasks schedulable *if* $U \leq n(2^{\frac{1}{n}} - 1)$
- Single-processor EDF: n tasks schedulable *if and only if* $U \leq 1$
- Useful numbers: $2^{1/2} = 1.414213562$; $2^{1/3} = 1.25992105$; $2^{1/4} = 1.189207115$; $2^{1/5} = 1.148698355$; $2^{1/6} = 1.122462048$; $2^{1/7} = 1.104089514$; $2^{1/8} = 1.090507733$.

Values of the Standard Normal Distribution (i.e. $F(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-0.5z^2} .dz$) for z>0

z	**.00**	**.01**	**.02**	**.03**	**.04**	**.05**	**.06**	**.07**	**.08**	**.09**
0.0	.5000	.5040	.5080	.5120	.5160	.5190	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7157	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7969	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8513	.8554	.8577	.8529	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9215	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9492	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998

[This page is intentionally (almost) blank – use as extra space]


I AM ELON TUSK
AND I APPROVE THIS EXAM
imgflip.com