CS143A: Principles of Operating Systems
Homework 1, Due Date:

Name: ANSWER KEY
ID: _____

Submission Instructions: Please submit a PDF file containing all answers to Gradescope. The following template should give you enough space for your answers and is already properly formatted. If you need more space for a problem, then you can add a page, but please do not have two problems on one page. (Note: Some problems are multiple parts – it is OK for these parts to be on the same page).

Question 1: Polling, Interrupts, Traps, and DMA

a) Alice is a student in the OS class, but she is having some trouble understanding the relationship and/or differences between some concept pairs. She asks you, her smart friend, to explain the the relationship and/or differences between the following:

i) [5] Synchronous I/O vs Asynchronous I/O

A program using synchronous I/O needs to wait for the I/O to finish before proceeding. A program using asynchronous I/O will not wait for the I/O to finish; it will ask for I/O and then continue running. When the I/O is ready, an interrupt will be generated to notify the program that the I/O is ready.

ii) [5] Interrupts vs Traps

An interrupt is generated either from the software or the hardware; it is used to inform the program of some kind of event (e.g., I/O). On the other hand, a trap is a special type of interrupt that is generated by the program to go into kernel mode.

b) [5] Alice was impressed by your responses to the above, so she asks you a few questions to test her understanding. However, at this point, you are a bit annoyed at this involuntary group work and decide to just place an **X** in the appropriate box to indicate whether each statement is True or False.

Statement	True?	False?
The overhead of polling depends on the polling frequency.	X	
A file I/O can be implemented using either traps or polling.	X	
Traps are software-generated interrupts.	X	
Polling is often a viable option for slow and asynchronous devices.		X
DMA usually slows down overall system performance.		X

c) Alice, after realizing that she's lost in the class, panics and decides to buy a laser printer in order to print out all the course notes. Suppose that she bought a laser printer that produces up to 45 pages per minute, where each page consists of 5000 characters. The manual for the laser printer states that the system uses interrupt-driven I/O by raising an interrupt for every character. Answer the following questions, showing all work.

i) [5] If each interrupt takes 50 microseconds to process, how much CPU time will be spent processing interrupts (in %)?

Answer: Percentage of CPU time spent: 18.75 %

Assume that you print 45 pages. Then, the ratio to find is: $\frac{time_{interrupts}}{time_{print\ 45\ pages}}$.

$time_{interrupts} = 45\ pages * \frac{5000\ characters}{1\ page} * \frac{50\ microsec}{1\ character} * \frac{1\ second}{10^6 microsec} * \frac{1\ minute}{60\ seconds} = 0.1875\ minutes$.

It is obvious that $time_{print\ 45\ pages} = 1\ minute$.

Therefore, the percentage of CPU time spent is 18.75%

ii) [5] Alice has an option of switching her laser printer's system to a polling-driven implementation instead. A polling-driven implementation takes up 15% of the CPU time to process I/O. Should Alice switch to using polling instead of interrupts? Explain why or why not.

Answer: Alice should use polling-driven I/O because it has less overhead (it takes up a smaller proportion of CPU time).

iii) [5] Now suppose Bob (a sneaky prankster) replaces Alice's laser printer with a toaster on April 1st. Unfortunately, Alice urgently needs to print lecture notes for the upcoming class, and decides that the easiest solution is to buy a new printer. The new printer has the same specifications as Alice's old printer: it can produce 45 pages per minute, where each page consists of 5000 characters. However, an upgrade in the system hardware allows the polling-driven implementation to take up only 10% of the CPU time instead. How long should the interrupt-driven implementation take for each interrupt to "break even"?

Answer: Time taken per interrupt: 26.666 microseconds

We use the same formula as in part (a). $10\% = \frac{time_{interrupts}}{time_{45\ pages}}$.

We know that $time_{45\ pages} = 1\ min.$ Let T be the time taken per interrupt. Then,

$time_{interrupts} = 0.1\ minutes = 45\ pages * \frac{5000\ characters}{1\ page} * \frac{T\ microsec}{1\ character} * \frac{1\ second}{10^6 microsec} * \frac{1\ minute}{60\ seconds}$.

Therefore, $0.1\ minutes = \frac{45 * 5000 * T}{10^6 * 60}\ minutes$.

Solving for T gives approximately 26.666 microseconds.

Question 2: Parallel, Distributed, and Real-time Systems

a) [12] Despite her efforts, Alice is confused in the next lecture and doesn't understand the difference between multiprogramming, multiprocessing and timesharing. However, she notices that other students have similar questions, and decides to wait for an answer by the professor. Unfortunately for her, at the exact moment the question was being answered, Bob turned off Alice's computer and she missed everything. You feel bad for Alice, and decide to explain the relationship between multiprogramming, multiprocessing, and timesharing below.

In multiprogramming, there is typically just one CPU and processes run on the CPU in turns. Timesharing is similar to multiprogramming in that we have one CPU and multiple processes, but as the name suggests, the CPU splits up time into equal size chunks, each of which is "given" to the process so that it can run. On the other hand, multiprocessing is more used in parallel systems -- there are multiple CPUs, and processes are able to run on different CPUs simultaneously.

b) [8] Later, Alice does not understand the concept of real-time systems, and is once again, asking you for help. You are tired of answering her questions, so for each of the following activities, you place an **X** in the appropriate box to indicate whether they are: i) hard real-time, ii) soft real-time, or iii) not real-time.

	Hard real-time	Soft real-time	Not real-time
Uploading Homeworks to Gradescope			X
Attending a Lecture via Zoom		X	
Using an intelligent braking control system for vehicles	X		
Watching a video on Youtube		X	

Question 3: System Calls

a) [6] Alice needs your help to do some system programming in C. She asks you, since you are an expert at using the *man* pages. Help Alice by writing the names of the system calls (without parameters) that ...

	System Call
… create a new process	fork
… change file permissions	chmod
… create a new file to save text	open

b) [10] Suppose that Alice wants to write a program that takes in a list of file names from the command line and outputs the total number of characters from all the given files to stdout. Alice asks you if the following system calls must be used at some point in the program. Indicate whether the system call is used or not used by placing an **X** in the appropriate box. Then, if the system call is used, then explain how in the "Explanation" column.

	Use	Not Used	Explanation
read()	X		You need to be able to read each of the files in order to count the number of characters in it.
wait()		X	(no explanation needed)
open()	X		You need to open a file before reading it.
getpid()		X	(no explanation needed)

c) [4] Alice takes a short coffee break after writing half of her program. Unfortunately, Bob notices that Alice is gone, and decides that he wants to mess around with Alice's computer. Bob writes a program that can only modify operations that are not privileged, and runs it on poor Alice's computer. Place an **X** in the appropriate box to indicate whether Bob's program is able to modify the operation.

	Yes	No
Count characters in a line	X	
Check list of currently running processes		X
Process hardware interrupt		X
Force system to sleep		X

Question 4: Processes and Threads

a) [4] Alice is very grateful for your help so far, but wants to study with you before the quiz next week. She believes that she will be tested on the process control block and its contents, so she asks you to indicate, by placing an **X** in the appropriate box, for whether each of the following are stored in the PCB.

	Stored?	Not Stored?
All available hardware devices		X
Program Counter	X	
Number of Processes		X
Process ID	X	

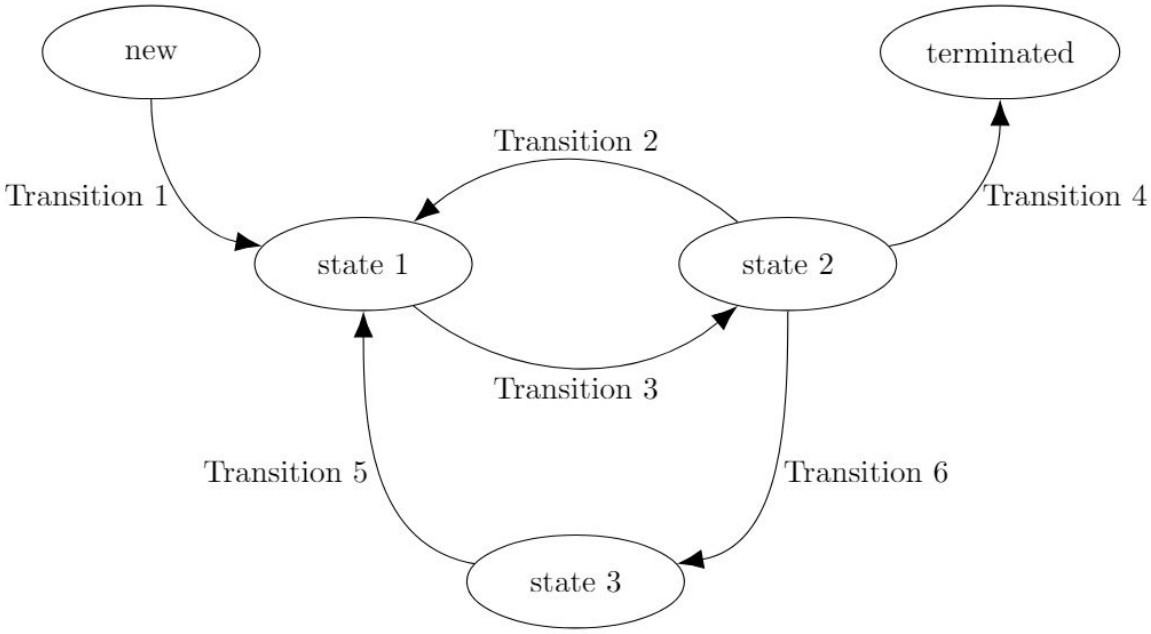
b) [4] Back to system programming! Alice is now tasked with writing a program to do matrix multiplication. However, Bob has once again replaced Alice's computer with a uniprocessor toaster behind her back. The program will be graded on speed, so she wishes to make the program run as fast as possible. Instead of trying to come up with a more efficient algorithm, Alice is determined to use either more threads (`pthread_create`) or more processes (`fork`) to improve performance. Should Alice use more threads or more processes (assuming that you can only pick one)? Explain why.

Answer: Alice should use threads. Two reasons why: Threads use fewer resources in context switching, they are able to share memory with each other. The matrix multiplication application inherently stores a lot of memory that does not necessarily need to be copied to a new process.

c) [4] Alice considers what you have said above, and then decides to use threads. However, she has a bug in her implementation because she has forgotten which of the following are shared among threads and which are not. Indicate which of the resources are shared among the threads and which are private by placing an **X** in the appropriate box.

	Shared?	Private?
Stack Memory		X
Signals	X	
Global Variables	X	
Program Counter		X

d) [18] To study for the midterm, Alice made a state diagram for the different states and state transitions that a process goes through in its lifetime. Unfortunately, when Alice was looking away, Bob erased most of the states and state transitions in the figure. Fill in the table at the bottom of the page, which corresponds with the state diagram.



	State
State 1	ready
State 2	running
State 3	waiting

	Transition
Transition 1	admitted
Transition 2	interrupt
Transition 3	scheduler dispatch
Transition 4	exit
Transition 5	I/O or event completion
Transition 6	I/O or event wait