Operating Systems Principles
First Quiz, Section 1
Friday, September 22nd, 2017
Instructor: Ghassan Shobaki

Student Name: ________________________ **Student Number:** ______________________

Q1. Modern operating systems are written in high-level languages (mainly C and C++). Give two good reasons for writing an operating system in a high-level language and explain each reason in the space provided.
(14 points)

Q2. In a time sharing OS, one process runs into an infinite loop. Answer the following. (12 points)
Does the OS detect infinite loops?

Does the OS terminate a process that runs into an infinite loop?

Will this cause other processes to hang? Why?

Q3. Compare the **monolithic** and the **loadable module** OS structures by giving a **clear** advantage of each structure relative to the other. Explain each advantage in the provided space. (14 points)
Advantage of monolithic relative to loadable modules

Advantage of loadable modules relative to monolithic

Q4. The table below shows three events that may cause a transition in a process's state. For each event, the table shows the state transition that immediately follows the event and the queue in which the process gets placed after the transition. Fill in the missing entries in the table. If there is no queue, write **none**. (30 points)

Causing Event	Transition From State	 To State	Queue after transition (if none, enter **none**)
System call to read from keyboard			
	Running	Ready	
Scheduler dispatch			

Q5. An OS running on a **single CPU** has the following two processes:

- P_1 consists of a 70-ms CPU burst followed by an I/O request followed by a 30-ms CPU burst
- P_2 consists of a 40-ms CPU burst followed by an I/O request followed by a 60-ms CPU burst

Draw a complete timing diagram showing how a time-sharing operating system will handle these processes under the following assumptions:

- Both I/O requests go to the same device and each request takes 80 ms.
- The I/O device will start processing a queued request as soon as the current request is done (no delay).
- Whenever P_1 and P_2 are **both** ready, the scheduler will give the CPU to the process that did not get the last CPU time (fair alternation if **both** are ready). In the beginning, the CPU is given to P_1.
- The time quantum is 50 ms
- The kernel time (including initiation of I/O requests, scheduling and context switching) is 10 ms.
- The OS starts each I/O request right at the beginning of the kernel time.
- Ignore any delay or transition time that was ignored in class. (30 points)

Show both the CPU and the I/O device in your diagram and make sure that you align the CPU time with the I/O time.