QUESTION 1

1. System calls are identical to library function calls in a high-level programming language such as C.

   - ○ True: For example, fopen() in C programming is a system call to open a file.
   - [✓ False: User programs can access services in the OS by invoking software functions implemented in the OS through system calls.]

0.15 points

QUESTION 2

1. Why is there no transition from the waiting state to the running state?

   - ○ Because a process in the waiting state is unable to execute even if it is given access to the CPU.
   - ○ Because once a process in the waiting state is ready to execute, it would first move to the ready state and then to the running state when scheduled for execution by the OS.
   - ○ Because this allows the OS to methodically select the process to execute from among all processes that are ready to execute on the CPU.
   - [✓ All the other options are correct.]

0.15 points

QUESTION 3

1. Where is a Process Control Block stored and who can access it?

   - ○ In the user memory space and can be accessed by the OS as well as the process.
   - ○ In the kernel memory space and can be accessed by the OS as well as the process.
   - [✓ In the kernel memory space and can be accessed only by the OS.]
   - ○ In the user memory space and can be accessed by only the OS.

0.15 points

QUESTION 4

1. What comprises a process context and where is it stored?

   - [✓ All registers required to execute/resume the process (program counter, stack pointer, memory protection registers, etc.) and it is stored in the PCB.]
   - ○ The process memory as defined by the base and limit register values and it is stored in the user memory space.
   - ○ All registers required to resume the process (program counter, stack pointer, memory protection registers, etc.) and it is stored in the user memory space.
   - ○ Only base and limit register values for a process and it is stored in the PCB.

0.15 points

QUESTION 5

1. Trap is an interrupt generated only due to software exceptions.
   - ○ True: CPU transfers control to the OS to process the exception.
   - [✓ ● False: CPU generates trap either due to software exception or system call, and transfers control to the OS.]
   - ○ False: OS generates trap due to software exceptions and transfers control to the CPU.
   - ○ False: OS generates trap and processes it.

0.15 points

QUESTION 6

1. When a user program tries to execute an I/O instruction directly,
   - ○ the CPU executes the instruction
   - ○ the CPU checks if the request is within range of the base and limit registers
   - [✓ ● the CPU checks for its validity and legality before executing it.]
   - ○ the CPU generates a trap and transfers control to the OS.

0.15 points

QUESTION 7

1. In real-time systems, average response time of user programs should be minimized.
   - ○ True: This will ensure that the required deadlines are met.
   - ○ True: This will ensure that the system is responsive.
   - [✓ ● False: Real-time systems require guaranteed satisfaction of deadlines, irrespective of average response time.]
   - ○ False: Real-time systems require average response times to be maximized.

0.15 points

QUESTION 8

1. Describe one advantage and one disadvantage of batch systems.
   - ○ Memory management is simple (1 user job); CPU utilization is not efficient (scheduling of jobs is complex).
   - [✓ ● Memory management is simple (1 user job); CPU utilization is not efficient (idling during I/O).]
   - ○ CPU utilization is very efficient (1 user job to schedule); Memory management is complex (deciding which user job to keep in memory).

0.15 points

QUESTION 9

1. What is an advantage and a disadvantage of one-to-one mapping of logical (user) and physical (kernel) threads?

- ○ More concurrency than many-to-one mappings; mapping is very complex.
- ○ More concurrency than many-to-many mappings; requires creation of several kernel threads (one per user thread).
- [✓ More concurrency than many-to-one mappings; requires creation of several kernel threads (one per user thread).]
- ○ Mapping is very simple; less concurrency than many-to-one mappings.

0.15 points

QUESTION 10

1. What are the key OS features needed for multiprogramming.

- ○ Multiprocessing, DMA and I/O protection.
- ○ CPU scheduling, multiprocessing and DMA.
- [✓ I/O device management, CPU scheduling and memory management.]
- ○ Memory management, I/O protection and CPU scheduling.

QUESTION 3

1. What is the disadvantage of many-to-one mapping between logical (user) and physical (kernel) threads?

- ○ The mapping process is very complex.
- [✓ A blocked user thread will block all the other user threads that are mapped to the same kernel thread.]
- ○ There is no disadvantage.
- ○ A blocked user thread will also block all the other kernel threads in the syste

QUESTION 7

1. What is an Operating System?

- ○ Software program that controls other user programs.
- ○ Allocates and manages hardware resources.
- ○ Always ready to accept new commands from users and hardware.
- [✓ All the other options are correct.]

QUESTION 1

1. There is one base and one limit register in the hardware for each user program.

   - ○ True: This is necessary to ensure memory protection for each user program.
   - [✓ False: There is only one base and one limit register overall in the hardware, which are loaded with appropriate values by the OS before executing the user program.]
   - ○ False: There is only one base and one limit register overall in the hardware, which are loaded with appropriate values by the CPU before executing the user program.
   - ○ False: These registers are in the OS, which also checks whether each memory access is within range.

0.15 points

QUESTION 2

1. The following sequence of steps precisely defines the processing of a system call

   - ○ OS generates a trap, OS switches to kernel mode, OS identifies the appropriate function for the system call and executes it, upon function completion OS switches the mode back to user mode and transfers control to the user program.
   - ○ CPU generates a trap, CPU switches to kernel mode, CPU identifies the appropriate function for the system call and executes it, upon function completion CPU switches the mode back to user mode and transfers control to the user program.
   - [✓ CPU generates a trap, CPU switches to kernel mode and transfers control to the OS, OS identifies the appropriate function for the system call and executes it, upon function completion OS switches the mode back to user mode and transfers control to the user program.]
   - ○ CPU generates a trap, CPU switches to kernel mode and transfers control to the OS, OS identifies the appropriate function for the system call and executes it, upon function completion CPU switches the mode back to user mode and transfers control to the user program.

0.15 points

QUESTION 3

1. Kernel or monitor or supervisor mode is the same as "superuser" in Linux.

   - [✓ False: Kernel mode is a hardware mode of operation, whereas "superuser" is the "root" user account.]
   - ○ True: Both, the kernel mode as well as "superuser" allows one to execute privileged instructions in hardware.
   - ○ False: "superuser" is more privileged than kernel mode.
   - ○ False: "superuser" is the same as supervisor mode, whereas kernel or monitor mode is more privileged.

0.15 points

QUESTION 5

1. What are the typical resource constraints of embedded systems?

   - ○ Small memory and low power (battery operated).
   - ○ Low network bandwidth and small displays.
   - ○ Slow processors
   - [✓ ● All the other options are correct.]

0.15 points

QUESTION 6

1. What is the difference between shared memory and message passing based Inter-Process Communication.

   - ○ They are two different names for the same communication mechanism which uses kernel memory space.
   - [✓ ● In message passing, the communication is handled by the OS; In shared memory, the communication is handled directly by the processes through a common memory space.]
   - ○ In shared memory, there is memory in the kernel space that is shared between the processes; In message passing, no memory is required in the kernel memory space.
   - ○ They are two different names for the same communication mechanism which uses user memory space.

0.15 points

QUESTION 7

1. What parts of the process memory space are shared between threads in the same process.

   - ○ Everything is shared between threads in a single process.
   - ○ Only Text and OS resources (files, etc.) are shared; all other parts are private to the threads.
   - [✓ ● Text, Data, Heap and OS resources (files, etc.) are shared between the threads; Stack and Registers are not.]
   - ○ Text, Data and Heap are shared between the threads; OS resources (files, etc.), Stack and Registers are not.

0.15 points

QUESTION 8

1. DMA does not use interrupts.

   - ○ False: DMA uses interrupts once the block transfer is complete (between memory and I/O device) to notify the CPU.
   - [✓ ● True: DMA bypasses the CPU and hence does not require interrupts.]

0.15 points

QUESTION 10

1. What are the two typical execution orders in fork() system call

   - ○ Forked child processes wait for parent process to terminate; Parent process can wait for the forked child processes to terminate before proceeding.
   - [✓ Parent process can execute in parallel to the forked child processes; Parent process can wait for the forked child processes to terminate before proceeding.]
   - ○ Parent process executes before forked child processes; Forked child processes execute before parent process can continue.

QUESTION 6

1. What is the difference between a long-term and medium-term scheduler?

   - ○ Long-term scheduler schedules processes in main memory on the CPU; Medium-term scheduler decides which active processes to keep in main memory and which ones to swap to disk when load is heavy.
   - ○ They are identical.
   - ○ Long-term scheduler decides which new processes should be brought to main memory from disk; Medium-term scheduler schedules processes in main memory on the CPU.
   - [✓ Long-term scheduler decides which new processes should be brought to main memory from disk; Medium-term scheduler decides which active processes to keep in main memory and which ones to swap to disk when load is heavy.]

QUESTION 6

1. What are the four memory regions of a process and how are they used.

   - ○ "Text" for documentation about the process; "Data" for all variables; "Stack" for managing function calls; "Heap" for garbage collection.
   - ○ "Code" for instructions; "Global" for static and global variables; "Function" for managing function calls and local function variables; "Dynamic" for dynamically created variables.
   - [✓ ◉ "Text" for instructions; "Data" for static and global variables; "Stack" for managing function calls and local function variables; "Heap" for dynamically created variables.]
   - ○ All the other options are correct.

QUESTION 7

1. How do you distinguish between direct and indirect message passing for inter-process communication?

   - [✓ ◉ In direct message passing, communication occurs between specific processes; In indirect message passing, any process with access to a mailbox can send/receive messages to/from that mailbox.]
   - ○ In direct message passing, communication occurs between specific processes through shared user space memory; In indirect message passing, communication occurs through mailboxes.
   - ○ Direct message passing is the same shared memory; Indirect message passing uses the kernel memory space.

QUESTION 8

1. Multiprocessing without multiprogramming is efficient?

   - ○ Yes, they are different techniques that can applied independently and all combinations are efficient.
   - ○ Yes, even without multiprogramming multiprocessing will ensure high CPU utilization.
   - [✓ ◉ No, without multiprogramming there will be only one user job to execute and hence CPU utilization will be low.]

QUESTION 1

1. What is the difference between a process and a program?

- ○ They are identical
- ○ Every user program can have at most one active process in the system.
- ○ They are two different and unrelated concepts.
- [✓ Process is an instance of a program that is currently active in the system]

0.15 points

QUESTION 3

1. What is the difference between multi-threading, multi-programming and multi-processing?

- ○ They are all different terms for the same concept of time-multiplexed execution.
- [✓ multi-threading and multi-programming are identical; multi-processing means multiple CPU cores for execution.]
- ○ multi-threading means multiple threads of execution within a process; multi-programming and multi-processing are identical.
- ○ multiple threads of execution within a process; multiple processes ready for execution in main memory; multiple CPU cores for execution.

0.15 points

QUESTION 4

1. Multiprogrammed systems allow parallel execution of jobs, hence they are also called time-shared systems.

- [✓ False: Parallel execution of jobs depends on the hardware (multiprocessing); multiprogramming allows efficient switching between jobs for CPU access (time-division access), and hence it is also called time-shared system.]
- ○ True: Multiprogramming allows simultaneous access to the CPU for multiple user jobs, and hence it is also called time-shared system.

0.15 points

QUESTION 5

1. In the process state transition diagram, what is a timer interrupt?

- [✓ It is an hardware interrupt generated by a hardware/software clock that tracks progress of physical time.]
- ○ It is an interrupt generated by the OS to denote process completion.
- ○ It is a trap interrupt generated by the CPU.
- ○ None of the options are correct.

0.15 points

QUESTION 7

1. What is an advantage and a disadvantage of many-to-many mappings of logical (user) to physical (kernel) threads?

   - ○ High concurrency and mapping is very simple many kernel threads would be required.
   - [✓ High concurrency and not many kernel threads required; mapping is very complex.]
   - ○ Mapping is very simple and not many kernel threads required; concurrency is low.
   - ○ High concurrency; mapping is very complex and many kernel threads would be required.

0.15 points

QUESTION 10

1. What is the difference between shared memory and message passing based Inter-Process Communication.

   - ○ They are two different names for the same communication mechanism which uses kernel memory space.
   - [✓ In message passing, the communication is handled by the OS; In shared memory, the communication is handled directly by the processes through a common memory space.]
   - ○ In shared memory, there is memory in the kernel space that is shared between the processes; In message passing, no memory is required in the kernel memory space.
   - ○ They are two different names for the same communication mechanism which uses user memory space.