Answer the following questions:-

**Question No. (1) [10 points] True/False**

1. Counting *semaphores* generalize the notion of a binary *semaphore*.  \( \text{T} \)
2. When a process is created using classical *fork* system call, process ID is inherited by the child process.  \( \text{F} \)
3. One example of a hardware solution to the critical section problem is a *monitor*.  \( \text{F} \)
4. A time-sharing system is a type of multiprogramming system.  \( \text{T} \)
5. The *layered kernel* approach imposes a strict ordering of subsystems such that subsystems at the lower layers are not allowed to invoke operations corresponding to the upper-layer subsystems.  \( \text{T} \)
6. By using the *system-call interface* it would be possible for the user to develop a new command interpreter.  \( \text{T} \)
7. Reliability is one of a CPU scheduling criteria.  \( \text{F} \)
8. If a user-level thread blocks for I/O, all the other user-level threads belonging to the same process have to block.  \( \text{F} \)
9. A multiprogramming system need interrupts to operate more efficiently.  \( \text{T} \)
10. User programs and system services interact in a *microkernel* architecture by using interprocess communication mechanisms (*IPC*) such as messaging.  \( \text{T} \)

**Question No. (2) [10 points] Multiple Choice**

1. The removal of process from active contention of CPU and reintroduce them into memory later is known as…
   - A. Interrupt
   - B. Swapping
   - C. Signal
   - D. Multiprogramming
2- The Purpose of Cooperating processes is …
   A. Information Sharing  
   B. Convenience  
   C. Computation Speed-Up  
   D. All of the above  

3- The degree of Multiprogramming is controlled by…
   A. CPU scheduler  
   B. Context switching  
   C. Long-term scheduler  
   D. Medium-term scheduler  

4- The producer-consumer problem refers to:
   A. Mutual-exclusion when n jobs sharing a resource  
   B. Multiple clients and servers accomplishing a computation  
   C. The need for synchronization in systems where many processes share a resource  
   D. None of the above  

5- In multithreaded programs, the kernel informs an application about certain events using a procedure known as ……
   A. Signal  
   B. Upcall  
   C. Event handler  
   D. Pool  

6- LWP is…
   A. placed between system and kernel threads.  
   B. placed between user and kernel threads.  
   C. common in systems implementing one-to-one multithreading models.  
   D. none of the above.  

7- CPU burst distribution is generally characterized as
   A. constant  
   B. linear  
   C. polynomial  
   D. exponential or hyper-exponential  

8- The primary difference between user-level threads and kernel threads is …
   A. User level threads do not use OS services via system calls, where kernel threads require system calls .  
   B. User level threads are independent of each other, whereas kernel threads can write into each other’s memory space .  
   C. User level threads require memory management where kernel threads do not .  
   D. None of the above.
9- *Thread-specific data* is data that…
   A. is not associated with any process.
   B. has been modified by the thread but not yet updated to the parent process.
   C. is generated by the thread independent of the thread's process .
   D. is copied and not shared with the parent process.

10- The purpose of *dual mode* of operation is to…
   A. Protect the OS from user programs.
   B. Protect the user programs from an OS.
   C. Protect one program from another user program.
   D. Protect the computer hardware.

Question No. (3) [20 points]

A. One set of operating system services provides functions that are helpful to the user. 
   Mention three of such services (Not Description).
   - Program execution
   - I/O operations
   - File-system manipulation
   - Communications
   - Error detection

B. Elaborate what is the meaning of a Process Control Block (PCB)?

Each process is represented in the operating system by a data structures called process control block (PCB).
The PCB contains all the Information about a process.
Information associated with each process
   - Process state
   - Program counter
   - CPU registers
   - CPU scheduling information
   - Memory-management information
   - Accounting information
   - I/O status information

C. For each of the following pairs of terms, define each term; make sure to clarify the key differences between the two terms
   i. Process and thread.

   ➢ *Thread is a program execution that uses the resources of process*
Thread is a basic unit of CPU utilization, which comprise a thread ID, program counter, register set, stack.

Thread shares with other threads belonging to the same process its code section, data section, and other OS resources (open files and signal).

A thread is used to achieve a reduction in switching overhead.

A process is a program in execution.

It is composed of:

- Text section (which is the program code).
- Data section (which contains global variables).
- Stack (which contains temporary data such as function parameters, return addresses, and local variables)
- Current activity (which is represented by the value of the program counter and the contents of the processor's registers).
- Heap (which is a memory that is dynamically allocated during process run time).

Uses of process to provide concurrency with an application incurs high process switching overhead.

ii. Direct and indirect communication

Direct communication

- each process that wants to communicate must explicitly name the recipient or sender of the communication
  - send \((P, \text{message})\) – send a message to process \(P\)
  - receive \((Q, \text{message})\) – receive a message from process \(Q\)

Properties of communication link

- Links are established automatically between every pair of process that want to communicate. The processes need to know only each other's identity to communicate.
- A link is associated with exactly two processes
- Between each pair of processes, there exists exactly one link

Two addressing mode
Symmetry in addressing
- Both sender and receiver must name the other to communicate

Asymmetry in addressing
- Only the sender names the recipient, the recipient is not required to name the sender.

**send** \((P, \text{message})\) – send a message to process \(P\)

**receive** \((\text{id, message})\) – receive a message from any process, \(\text{id}\) is set to the name of the sender with which communication has taken place

Disadvantages in direct communication (symmetry & Asymmetry)
- Changing the id of a process may necessitate examining all other process definitions.

**Indirect communication**

- Messages are send to and received from mailboxes (also referred to as ports)
  - Mailbox is an *object into which messages can be placed and from which messages can be removed.*
  - Each mailbox has a unique id
  - Two processes can communicate *only if they have share a mailbox*

- Properties of indirect communication link
  - A link established between pair of processes *only if both processes have a shared mailbox*
  - A link may be *associated with many processes*

Between each pair of communicating processes, there may be a number of *several communication links, with each link corresponding to one mailbox*

- A mailbox may be owned either by a process or by the operating system
- If the mailbox is owned by a process, then we distinguish between the owner (who can only receive messages through this mailbox) and the user (who can only send messages to the mailbox)
  - When a process that owns a mailbox terminate the mailbox terminate and disappear.
  - Any process that subsequently sends a message to this mailbox must be notified that the mailbox no longer exists.
A mailbox that is owned by the operating system has an existence of its own. It is independent and is not attached to any particular process.

The OS must provide a mechanism that allows a process to do:
- create a new mailbox
- send and receive messages through mailbox
- destroy a mailbox

E. Draw the state diagram of a process from its creation to termination, including all transitions, and briefly elaborate every state and every transition.

- As a process executes, it changes state
  - new: The process is being created
  - running: Instructions are being executed
  - waiting: The process is waiting for some event to occur (e.g., I/O)
  - ready: The process is waiting to be assigned to a processor (CPU)
  - terminated: The process has finished execution

F. What are the three main reasons for aborting (terminating) a child process?
   1. The child has exceeded its usage of some of the resources that it has been allocated.
   2. The task assigned to the child is no longer required.
   3. The parent is exiting.

Question No. (4) [15 points]

A. Elaborate all major differences between a semaphore wait/signal and a condition variable wait/signal.
B. Define the meaning of a race condition? Illustrate your answer with an example.

A race condition is a situation in which more than one processes or threads access a shared resource concurrently, and the result depends on the order of execution.

Example (any example):

- Suppose two or more asynchronous processes or threads access to a common, shared piece of data.
- Any process can be preempted at any time
  - Your current bank balance is $1,000.
  - Withdraw $500 from an ATM machine while a $5,000 direct deposit is coming in.

<table>
<thead>
<tr>
<th>Withdrawal</th>
<th>Deposit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read account balance</td>
<td>Read account balance</td>
</tr>
<tr>
<td>Subtract 500.00</td>
<td>Add 5000.00</td>
</tr>
<tr>
<td>Write account balance</td>
<td>Write account balance</td>
</tr>
</tbody>
</table>

Consider the possible outcomes:
$5500 (correct answer if there is no race)

The proper outcome, of course, is that your checking account will have $5,500 after both operations are finished. Either the Deposit sequence of operations takes place first or the Withdrawal sequence does. Either way, the answer is the same.

$500

Suppose that the system just completed step 1 of the Withdrawal operation. At that time, the thread gets preempted and the Deposit thread gets to run. It goes through its steps: reads the account balance ($1,000), adds $5000 ($6,000), and writes the result ($6,000). Then the Withdrawal thread is allowed to resume running. It already completed step 1, where it read $1,000 (not knowing that the value has been modified!), so it resumes operation with step 2 and subtracts $500 from $1,000,
storing $500 in step 3. Your account balance is now $500 — the actions of Deposit have effectively vanished.

$6,000

Now suppose that the operations are now scheduled in a slightly different order. The Deposit thread runs first, reads the account balance ($1,000) and adds $5,000 (giving $6,000). At this time, the thread is preempted and the Withdrawal thread is scheduled to run. In step 1, it reads the balance ($1,000, since the new balance was not yet written), subtracts $500, and stores the result, $500. When the Deposit thread is now given a chance to run, all it has left to do is step 3: store its result, $6,000. The effects of the Withdrawal thread have been obliterated.

A. What is the critical section? A good solution to the critical section problem must satisfy three conditions, mutual exclusion, progress and bounded waiting. Explain the meaning of each condition? Does starvation violate the progress condition?

Each process has a segment of code, called a critical section, where the process may change common variables, updating a table, writing a file, and so forth.

**Mutual Exclusion** - If process P, is executing in its critical section, then no other processes can be executing in their critical sections.

**Progress** - If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely – deadlock-free condition.

**Bounded Waiting** - A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted – starvation-free condition.

The progress condition only guarantees the decision of selecting one process to enter a critical section will not be postponed indefinitely. It does not mean a waiting process will enter its critical section eventually. Therefore, starvation is not a violation of the progress condition. Instead, starvation violates the bounded waiting condition.
Consider the following set of processes (P₁, P₂, P₃, P₄, and P₅), with the length of the CPU burst time given in milliseconds:

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival time</th>
<th>CPU Burst</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>P₂</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>P₃</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>P₄</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>P₅</td>
<td>7</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Fill the entries of the following table with waiting time, average waiting time and average turnaround time for each indicated scheduling algorithm and each process. Ignore context switching overhead. A smaller value means a higher priority. *(Show your work)*

<table>
<thead>
<tr>
<th>Scheduling Algorithm</th>
<th>Waiting Time</th>
<th>Average Waiting Time</th>
<th>Average Turnaround Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P₁</td>
<td>P₂</td>
<td>P₃</td>
</tr>
<tr>
<td>Non Preemptive Shortest Job First</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preemptive Priority</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Round Robin (time quantum =4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Come First Served</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ans.

<table>
<thead>
<tr>
<th>Preemptive Priority</th>
<th>Process</th>
<th>Start time</th>
<th>Finish time</th>
<th>Waiting</th>
<th>turnaround</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>16</td>
<td>23</td>
<td>15</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>14</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>14</td>
<td>16</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>7</td>
<td>13</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>6.2</td>
<td>10.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## FCFS

<table>
<thead>
<tr>
<th>Process</th>
<th>Start time</th>
<th>Finish time</th>
<th>Waiting</th>
<th>turnaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>10</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>15</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>17</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>23</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>6</strong></td>
<td><strong>10.6</strong></td>
<td><strong>0</strong></td>
<td><strong>5.3</strong></td>
</tr>
</tbody>
</table>

**First Come First Serve Gantt Chart:**

![Gantt Chart](image1.png)

## Non-SJF

<table>
<thead>
<tr>
<th>Process</th>
<th>Start time</th>
<th>Finish time</th>
<th>Waiting</th>
<th>turnaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>23</td>
<td>15</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>10</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>16</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>4.4</strong></td>
<td><strong>9</strong></td>
<td><strong>0</strong></td>
<td><strong>6</strong></td>
</tr>
</tbody>
</table>

**Shortest Job First Gantt Chart:**

![Gantt Chart](image2.png)

## RR with Q=4

<table>
<thead>
<tr>
<th>Process</th>
<th>Start time</th>
<th>Finish time</th>
<th>Waiting</th>
<th>turnaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>20</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>21</td>
<td>13</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>13</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>23</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>8.4</strong></td>
<td><strong>13</strong></td>
<td><strong>0</strong></td>
<td><strong>8.8</strong></td>
</tr>
</tbody>
</table>

**Round Robin Gantt Chart:**

![Gantt Chart](image3.png)
<table>
<thead>
<tr>
<th>Scheduling Algorithm</th>
<th>Waiting Time</th>
<th></th>
<th></th>
<th></th>
<th>Average Waiting Time</th>
<th>Average Turnaround Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P₁</td>
<td>P₂</td>
<td>P₃</td>
<td>P₄</td>
<td>P₅</td>
<td></td>
</tr>
<tr>
<td>Non Preemptive Shortest Job First</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>4.4</td>
</tr>
<tr>
<td>Preemptive Priority</td>
<td>0</td>
<td>15</td>
<td>6</td>
<td>10</td>
<td>0</td>
<td>6.2</td>
</tr>
<tr>
<td>Round Robin (time quantum =4)</td>
<td>0</td>
<td>12</td>
<td>13</td>
<td>7</td>
<td>10</td>
<td>8.4</td>
</tr>
<tr>
<td>First Come First Served</td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>11</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

With my best wishes

*Dr. Walid Osamy*