

## Question 1

- a) List and describe five (5) ways a Java thread can move into state “**Ready to Run**”. Be sure to indicate the previous state and the method or environment change that causes the thread to move from that state to the “Ready to Run” state.
- b) Briefly describe the XA-Core system. Explain what it does, why it is a real-time system, whether it is a hard real-time or soft real-time system, and identify the characteristic that differentiates it from the other real-time systems discussed in this course.
- c) Unbounded Priority Inversion:
  - i) When does **unbounded priority inversion** occur?
  - ii) Name two techniques for dealing with unbounded priority inversion.
  - iii) Describe **one** of the techniques for dealing with unbounded priority inversion that you named in your answer to part ii).

## Question 2

For each part of this question, you are asked to draw a timing diagram. For each diagram you need to show the client and the server, plus any other processes specifically mentioned in each case. Ensure that you clearly label each message with the TFTP packet type or opcode, block number (if applicable), error code (if applicable), data size (if applicable), and sending/receiving TIDs. Show the entire transfer for each example. Ensure that your timing diagram follows the TFTP specification. If your timing diagram is dependent on any design decisions peculiar to your implementation (i.e. not required by the specification), explain those decisions along with your diagram.

- a) In this transfer, a client requests to write text file “readme.txt” to the server. “readme.txt” contains 600 bytes. After processing the first block of data, the server receives a packet from a different process using TID 999 (not the client’s TID) on the TID it is using for this exchange.
- b) Here, a client is reading binary file “logo.jpg” from the server. This file is 2048 bytes long. The third data packet is duplicated (by the network) during this transfer.
- c) A client requests to read text file “info.txt” from the server, but the file does not exist on the server.
- d) A client requests writing a 750 byte binary file “this.jpg” to the server, and file “this.jpg” already exists on the server and is read only.
- e) Finally, a client is writing text file “mytextfile.txt” to the server’s “A” drive. This file is 5000 bytes long. The second ACK packet is delayed and arrives after normal reception of the third ACK packet. When attempting to write the fifth block of data, the disk becomes full.

### Question 3

The original specification of the TFTP (RFC 783) contained a serious bug.

- a) What does **TFTP** stand for?
- b) What is the name given to the bug in the original TFTP specification and why is this name used?
- c) How is this bug corrected in RFC 1350?
- d) When writing an implementation of the TFTP:
  - 1) What must be done to ensure the bug does not occur?
  - 2) What else can be done with regards to the retransmission of packets?

### Question 4

For each of parts (a) through (c), one or more of the statements is correct. Place an "x" or a check mark in the [ ] beside each correct statement.

(a) A newly created Java thread starts executing:

- [ ] immediately after the `Thread()` constructor returns, regardless of the thread's priority.
- [ ] immediately after the `Thread()` constructor returns, as long as the new thread has higher priority than the thread that created it.
- [ ] sometime after the thread's `start()` method is invoked, when the Java Virtual Machine (JVM) allocates the processor to the thread.
- [ ] sometime after the thread's `run()` method is invoked, when the JVM allocates the processor to the thread.
- [ ] only after the thread that created the new thread terminates.

(b) A Java thread of a given priority cannot terminate until all higher priority threads terminate.

- [ ] True.
- [ ] False.

(c) Priority inheritance

- [ ] means that a thread inherits the priority of the thread that created it.
- [ ] is a technique for solving the unbounded priority inversion problem.
- [ ] specifies the order in which superclasses are inherited by a subclass when multiple inheritance is used in a Java program.
- [ ] may cause a thread's priority to be different from its priority as set via the `setPriority()` method.

### Question 5

- a) List two techniques that can be used for **testing** the components of a real-time concurrent system.
- b) What is the difference between **verification** and **validation**?
- c) What is the name of the **development process** recommended for real-time systems?
- d) How does the Heisenburg principle apply to real-time concurrent system testing?

### Question 6

- (a) The formula for the Liu/Layland utilization test is:  $\sum_{i=1}^N \left( \frac{C_i}{T_i} \right) < N(2^{1/N} - 1)$

where  $N$  is the number of processes,  $C_i$  is the computation time of the  $i$ 'th process, and  $T_i$  is the period of the  $i$ 'th process. The following table shows the utilization bound (as a percentage) for small values of  $N$ .

$N$	Utilization bound (%)
1	100.0
2	82.8
3	78.0
4	75.7
5	74.3
10	71.8

Consider the following process set:

Process	Period, $T$	Execution time, $C$
P1	100	7
P2	50	3
P3	25	6
P4	20	9
P5	10	1

- (i) Does this process set pass the Liu/Layland utilization test? Explain your answer.
- (ii) If the processes are assigned priorities using rate monotonic assignment, will the processes meet their deadlines? Explain your answer by drawing timelines for the processes, clearly showing when they are executing and when they are ready-to-run, but preempted.

(b) Consider the following process set:

Process	Period, $T$	Computation time, $C$
A	20	6
B	20	8
C	40	4
D	40	3
E	80	8

Is it possible to construct a cyclic executive to schedule these processes? Draw a timeline to explain your answer.