UOW 2-line logo_NEW LOGO IIT-logo

**INFORMATICS INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF COMPUTING**

**Session 2011-2012 (Re-sit Paper)**

Code: ECSI603 Level 6

Date:

Time:

Duration: 2 hours

**INSTRUCTIONS TO CANDIDATES**

Answer THREE questions.

Each question is worth 33 marks.

**Question 1**

1. Describe the features of a java monitor. [8 marks]
2. Explain why a java monitor considered as *re-entrant?*  [5 marks]
   1. Give a suitable java code example to illustrate a java monitor used for controlling access to a shared resource. [12 marks]
   2. Explain how mutual exclusion is achieved with respect to the code example given in part (i) [8 marks]

**Question 2**

1. Explain what is meant by a *process*? What methods do most operating system provid for communication between processes? [7 marks]
2. Explain the following multi-threaded programming issues:
   1. Interference [5 marks]
   2. Synchronisation [2 marks]
   3. Fairness [5 marks]
   4. Deadlock [2 marks]
3. Explain each of the following terms with respect to the interaction of threads with the main memory, and thus with each other.
   1. use [2 marks]
   2. assign [2 marks]
   3. load [2 marks]
   4. store [2 marks]
   5. read [2 marks]
   6. write [2 marks]

**Question 3**

Two workers are assigned the task of ironing clothes in a Launderette. There is only one ironing board and each worker has to take turns to use the ironing board.

1. Using Finite State Process (FSP) define three processes to model the two workers and the IroningBoard [14 marks]
2. Using your two processes define a composite process that models the complete system [4 marks]
3. Briefly explain how you have ensured mutual access to the shared resource with respect to the composite process [15 marks]

**Question 4**

Given the following FSP definition of three processes:

ITCH = ( scratch -> STOP ) .

CONVERSE = ( think -> talk -> CONVERSE ) .

|| CONVERSE\_ITCH = ( ITCH || CONVERSE ) .

1. Briefly explain the following constructs [6 marks]
   1. ||
   2. ->
   3. STOP
2. Write the process alphabet for each of the three processes [12 marks]
3. Draw the *Labelled Transition System Graph* (LTS) the following [8 marks]
   1. ITCH
   2. CONVERSE
4. Draw the *Trace Tree* for the following [7 marks]
   1. ITCH
   2. CONVERSE

**Question 5**

1. In Java programming language:
   1. What are the methods of implementing a thread class? [4 marks]
   2. With respect to (i) what are the most critical methods that you need to implement to make the threads do the work? [2 marks]
   3. With respect to your answer for (ii) write sample java code listing of the method for a thread that will need to work continuously. [6 marks]
2. With reference to code listing given in Appendix A, write a possible sequence of output. [5 marks]
3. Describe and explain the states in the thread life-cycle that will cause a thread to pause or break its execution. [16 marks]

**Question 6**

1. Describe the features of a java *semaphore*. [8 marks]
2. What are the advantages of using a java monitor over a semaphore? [7 marks]
   1. Give a suitable java code example to illustrate a java semaphore used for strict interleaving of actions of two concurrent actions [12 marks]
   2. Explain how mutual exclusion is achieved with respect to the code example given in part (i) [6 marks]

**Appendix A - Code for Question 5**

|  |  |
| --- | --- |
| 1. | class SimpleThread extends Thread |
| 2. | { |
| 3. | public SimpleThread(String str){ |
| 4. | super(str); |
| 5. | } |
| 6. |  |
| 7. | public void run() { |
| 8. | for (int i = 0; i < 10; i++) { |
| 9. | System.out.println(getName() + ": " + i); |
| 10. | try { |
| 11. | sleep((int)(Math.random() \* 1000)); |
| 12. | } |
| 13. | catch (InterruptedException e) {} |
| 14. | System.out.println(getName()); |
| 15. | } |
| 16. | } |
| 17. | } |
| 18. |  |
| 19. | class TwoThreadsTest |
| 20. | { |
| 21. | public static void main (String args[]) { |
| 22. | Thread firstThrd ; |
| 23. | Thread secondThrd ; |
| 24. | firstThrd = new SimpleThread("FirstThread") ; |
| 25. | secondThrd = new SimpleThread("SecondThread") ; |
| 26. | secondThrd.setPriority(10); |
| 27. | firstThrd.start(); |
| 28. | secondThrd.start(); |
| 29. | } |
| 30. | } |