


# National University of Computer and Emerging Sciences, Lahore Campus

|   |             |                          |              |           |
|---|-------------|--------------------------|--------------|-----------|
|  | Course:     | Advance Operating System | Course Code: | CS-505    |
|   | Program:    | MS(Computer Science)     | Semester:    | Fall 2016 |
|   | Duration:   | 180                      | Total Marks: | 80        |
|   | Paper Date: | 14 December, 2016        | Weight:      | 50%       |
|   | Section:    | ALL                      | Page(s):     | 3         |
|   | Exam:       | Final                    | Roll No.     |           |

**Instructions/Notes:** Use less text and more diagrams for explanation. Fill the answers in the first few pages of your answer sheet. Do the rough work on last pages or separate sheet, and do not do any cutting where you answer the questions. Number the answers as the questions are numbered. Answer all parts of one question, then move to another question. Do not distribute parts of a question on disparate pages. **Not following the instructions will cause to deduct marks.**

**Question 1 (5 points):** What is 2-Phase locking, and what is the difference between (use diagrams)

- A 2 Phase Locking
- B Strict 2 Phase Locking
- C Strongly Strict 2 Phase Locking

**Solution:** 2 phase locking has growing phase and shrinking phase. In strict 2 phase locking **write locks** never get freed until the transaction commits. In Strongly strict 2 phase locking, no lock is released before transaction commits. The diagram of 2PL is given in the book, there will slight difference in diagram of S2PL and SS2PL.

**Question 2 (10 points):** Recall Triple Modular Redundancy (TMR). There are three components A, B and C. Component A adds two integers, component B uses the output of A and multiplies it by 2. Component C uses the output of B and divided it by 2.

- A Apply Triple Modular Redundancy to the scenario. Use the values 100 and 2 for the input of A and see what is the final output. Draw diagram for showing inputs and outputs.
- B Now assume one copy of each component is faulty, the fault is that the component adds 1 to the correct answer. Show how the result is modified, if modified at all? Use an example with diagrams to prove your argument.

**Solution:** The diagram of TMR for three components is given in the book. For part A student just has to draw that diagram and put the labels how the inputs and outputs travel from one component to other.

For part B, student can assume any of the three copies of each component to be faulty. Due to majority voting there will be no effect on the output.

**Question 3 (5 points):** A real time system is scheduled according to rate monotonic scheduling. Following table shows the repetitive arrival times of each task. Now assign priority to each task, and give reason for that prioritization.

| Task | Arrival Times |
|------|---------------|
| 1    | 10            |
| 1    | 20            |
| 1    | 30            |
| 1    | 40            |
| 1    | 50            |
| 2    | 20            |
| 2    | 40            |
| 2    | 60            |
| 2    | 80            |
| 2    | 100           |
| 3    | 40            |
| 3    | 80            |
| 3    | 120           |
| 3    | 160           |
| 3    | 200           |

**Solution:** RMS gives highest priority to the process with shortest period. So process 1 gets priority 1, process 2 gets 2, and process 3 gets 3.

**Question 4 (15 points):** If all processes in above mentioned example use 10 execution cycles and they use the priorities you just assigned, then

- A argue whether we have missed any deadlines or not? If not, then prove, if yes, then tell which ones.
- B If deadlines are missed, then can you come up with an algorithm which allows all deadlines to meet? If not, then give reasons, if yes, then provide the algorithm.
- C If you cannot come up with the algorithm, then what is the solution, how to meet all deadlines?

**Solution:** No deadlines are missed as there are no deadlines mentioned. No need to discuss any new algorithm.

**Question 5 (10 points):** There are three distributed processes  $P_1, P_2$  and  $P_3$ , and three resources available to these processes  $R_1, R_2$  and  $R_3$ . All are running at the same time and try to acquire the resources in following manner.

- $P_1: R_1, R_2 R_3$
- $P_2: R_2, R_3 R_1$
- $P_3: R_3, R_1 R_2$

The processes are connected in a token ring configuration, meaning the token starts from  $P_1$  it tries to acquire a resource, then the token passes to  $P_2$ , then  $P_2$  tries to acquire a resource. Then the token passes to  $P_3$  and it tries to acquire the resource and passes the token back to  $P_1$ . In this way the execution continues. Based on this information do following

- A You could use ring algorithm here, instead use a **centralized resource manager** to manage the resources. Show the status of all **queues** in the manager at **each step** until the first deadlock occurs. Draw the resource allocation graph of the situation.
- B Use **wait-and-die** algorithm to resolve the first deadlock and draw resource allocation graph again. The arrival sequence of processes is  $P_1, P_2$  and then  $P_3$ .

**Solution:**  $P_1$  will acquire  $R_1, P_2$  will acquire  $R_2, P_3$  will acquire  $R_3$ , in second cycle  $P_1$  will try to acquire  $R_2, P_2$  will try to acquire  $R_3$  and  $P_3$  will try to acquire  $R_1$ , which will result into a deadlock. **CLM** will have three queues one for each resource and they will show requests pooled by each process.

In part B,  $P_3$  will die as it is waiting for a resource acquired by an older process ( $P_2$ ).

**Question 6 (5 points):** Instead of using a centralized manager use Chandi-Misra-Haas Algorithm for the deadlock detection. You do **not** need to use wait-and-die algorithm once again for deadlock removal, rather you only have to show the steps how the first deadlock is detected using Chandi-Misra-Haas Algorithm. **NOTE:** Draw diagrams for the explanation.

**Solution:** In second cycle when the deadlock will occur, any process  $P_1, P_2$  or  $P_3$  can start the query process, and in the end it will know that the deadlock has occurred.

**Question 7 (10 points):** There are 8 processes in a configuration, numbered from 1 to 8. The leader is the highest numbered process. After a while the leader crashes. Process number 1 holds the election, and configuration elects a new leader using **bully** algorithm. Draw a diagram of each step of leader election, including the respective messages and their types.

**Solution:** Bully algorithm uses three messages **election**, **OK**, and **coordinator** messages. The process will start from 1 through 7, holding 7 elections, process 7 will reply OK in all cases, but nobody will reply OK to his election, which will make process 7 as a leader. So process 7 will send a **coordinator** message.  $6 \times 2 + 1 + 1 = 14$  diagrams should be drawn to show each step.

**Question 8 (10 points):** Just like in the above given scenario, when the leader crashes, the configuration holds an election based on **ring** algorithm. This time process 1 and 4 start the election simultaneously. Draw the diagram of each step of leader election, including the messages and their types.

**Solution:** In ring algorithm, two tokens will be floated on the network by process number 1 and 4. Both will receive their tokens back and announce that the leader is process number 8. **Two diagrams** should be drawn, one for the election started by process 1, and second for the election started by process 4.

**Question 9 (10 points):** Two processes  $P_1, P_2$  having their own caches use bus snooping to keep their caches consistent, by using write-once protocol. In the beginning their caches are empty. Following is the sequence of actions performed by different processes. You have to tell the states of all caches in result of the actions (use diagrams)

- 1 Process  $P_1$  reads a word  $W_1$  from memory
- 2 Process  $P_2$  reads a word  $W_1$  from memory
- 3 Process  $P_2$  changes the word  $W_1$
- 4 Process  $P_1$  changes the word  $W_1$
- 5 Process  $P_2$  reads a word  $W_1$  from memory

**Solution:** First read will be fetched from the memory and the copy will be clean. Second read will also be from memory and the copy will be clean.. Third step makes the copy **dirty** on  $P_2$  and **invalid** at  $P_1$ . Forth step will make the copy **invalid** on  $P_2$  and **dirty** on  $P_1$ . Fifth step, changes any status  $W_1$  at  $P_1$  as **invalid**,  $P_1$  provides a copy to  $P_2$ . Now  $P_2$  has the **dirty** copy of  $W_1$ .