Department of Computer Science
COS226 - Concurrent Systems

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS 226: Class Test 1

Date: 14 August 2020
Marks: 20 marks

Instructions

1. The class test comprises of **5** questions on **4** pages.
2. **1** hour has been allocated for you to complete this class test. An additional 25 minutes has been allocated for you to download the paper and upload your answer document.
3. This assessment is subject to the University of Pretoria Integrity statement provided below.
4. You are allowed to consult any literature.
5. You are **not** allowed to discuss the questions with anyone.
6. Write your answers in a separate document (eg. using a word processor or handwritten and scanned) and submit the document as a **single PDF** attachment. An upload slot will be open on the ClickUP module page under the **Tutorial 1 - Question paper and upload** menu option for the duration of the tutorial class (15:30 to 16:20), and then for an additional 25 minutes to give you enough time to download this paper and upload your PDF. The submission deadline is therefore **16:45**. **No late submissions will be accepted**.
7. Please make sure that your **name and student number are clearly visible** in the document you upload.
8. **Marks per question**

Question:	1	2	3	4	5	Total
Marks:	4	2	4	6	4	20

Integrity statement:
The University of Pretoria commits itself to produce academic work of integrity. I affirm that I am aware of and have read the Rules and Policies of the University, more specifically the Disciplinary Procedure and the Tests and Examinations Rules, which prohibit any unethical, dishonest or improper conduct during tests, assignments, examinations and/or any other forms of assessment. I am aware that no student or any other person may assist or attempt to assist another student, or obtain help, or attempt to obtain help from another student or any other person during tests, assessments, assignments, examinations and/or any other forms of assessment.

1. Describe the two classes of properties that a concurrent program must satisfy for it to be deemed correct? (4)

2. What is the main difference between deadlock-freedom and starvation-freedom? (2)

3. Suppose you have been requested to upgrade your company's uniprocessor system to an n-way multiprocessor. How many processors would you need to achieve a 5-fold speedup, given that your company's program is 85% parallelizable? (4)

   Show your working out. You will not be awarded full marks if you only provide the final answer.

4. Consider the code below. Assume that a locking mechanism has been applied to all critical sections of the `Counter` class.

```
// File: Counter.java
public class Counter {
  private int value;

  public Counter(int val) { value = val; }

  public int getValue() { return value; }

  public void increment() {
    // acquire lock
    ++value;
    // release lock
  }

  public void decrement() {
    // acquire lock
    --value;
    // release lock
  }
}
```

```
// File: CounterExample.java
public class CounterExample  {
    static final Counter c = new Counter(-1);

    public static void main(String[] args)    {
        Thread thread1 = new Thread(new Runnable() {
            public void run() {
                int val;
                do {                    // event a_0^j
                    c.increment();      // event a_1^j
                    val = c.getValue(); // event a_2^j
                } while (val < 2);      // event a_3^j
            }
        });
```

```
        Thread thread2 = new Thread(new Runnable() {
            public void run() {
                int val;
                do {                        // event b_0^k
                    c.decrement();          // event b_1^k
                    val = c.getValue();     // event b_2^k
                } while (val > -2);         // event b_3^k
            }
        });

        thread1.start();
        thread2.start();
    }
}
```

(a) The statements below describe the order of execution for the events highlighted in the code (i.e. $\{a_0^j, \ldots, a_3^j, b_0^k, \ldots, b_3^k\}$, where j and k are loop iterations of `thread1` and `thread2`, respectively). Each statement consists of multiple events. Write down the value of the `Counter` object after each statement has executed. Assume that the `Counter` state continues across statements. (4)

	Statement	`Counter` **value**
(i)	$(a_0^0, a_3^0) \rightarrow (a_0^1, a_3^1)$	
(ii)	$(b_0^0, b_3^0) \rightarrow (b_0^1, b_3^1)$	
(iii)	$(a_0^2, a_3^2) \rightarrow (a_0^3, a_3^3) \rightarrow (a_0^4, a_2^4)$	
(iv)	$(b_0^2, b_2^2) \rightarrow a_3^4 \rightarrow b_3^2$	

(b) Suppose that a thread is in one of the following states at any given point in time: (2)

- TERMINATED: thread has exited
- BLOCKED: thread is temporarily inactive (waiting)
- RUNNABLE: thread is running or ready to run

In what states are threads `thread1` and `thread2` after the execution of the last statement (iv) in sub-question (a) above?

5. Mr X has decided to give some of his riches away as part of a philanthropic movement. He instructs his bankers to take money from one of his bank accounts and to randomly distribute it to the public. A new developer at the bank has been tasked with implementing the program on the bank's distributed system. Study the `luckyPayouts` function below and answer the questions that follow.

   Assume that all relevant variables and functions have been declared/initialized/implemented accordingly.

```
1  public void luckyPayouts(int fromAccountId, double payment, int
       numPayouts) {
2
3    for (int i = 0; i < numPayouts; i++)
4    {
5      // pick random bank account ID
6      int toAccountId = getRandomBankAccountId();
7
8      // transfer money as long as there is money in the 'from'
           account
9      if (accounts[fromAccountId].balance >= payment)
10     {
11       accounts[fromAccountId].balance -= payment;
12       accounts[toAccountId].balance   += payment;
13     }
14   }
15 }
```

(a) Identify the lines of code that form the critical section. Only specify **line numbers**. (2)

(b) Give reasons for your line selections above. (1)

(c) In the case of a lock being used to enforce mutual exclusion, explain why the critical section should be encapsulated in a try-finally block? (1)