The University of Nottingham
Malaysia Campus

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, SPRING SEMESTER 2011 - 2012

CONCEPTS OF CONCURRENCY

Time allowed TWO hours

---

*Candidates may complete the front cover of their answer book and sign their desk card but must **NOT** write anything else until the start of the examination period is announced.*

Answer question ONE and THREE other questions

All questions carry equal weight; the marks available for the sections of each question are shown in brackets on the right-hand side.

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn examination paper over until instructed to do so

1. (**Compulsory**) Select the correct answer for each question from the four options given, for each **incorrect answer 0.5 marks** will be deducted.

(a) "Only one process may use a resource at a time" best describes:

(i) circular wait;
(ii) mutual Exclusion;
(iii) hold and wait;
(iv) no pre-emption. [3 marks]

(b) Round Robin scheduling is essentially the preemtive version of:

(i) Shortest job First;
(ii) Priority scheduling;
(iii) First Come, First Served;
(iv) Multiple queues. [3 marks]

(c) A technique that accelerates the throughput of the system by efficiently using the CPU time is known as:

(i) preemptive scheduling;
(ii) time-sharing;
(iii) multiprocessing;
(iv) multiprogramming. [3 marks]

(d) Atomic actions:

(i) always lock memory;
(ii) can't be interleaved;
(iii) divisible and execute concurrently;
(iv) always cause a process switch. [3 marks]

(e) A critical section is a program segment which:

(i) should run in a certain specified amount of time;
(ii) where shared resources are accessed;
(iii) avoids deadlocks;
(iv) is used only for synchronization. [3 marks]

(f) Within a monitor, calling the **wait** monitor operation:

(i) puts the calling thread to sleep, until another thread calls signal;
(ii) wakes one thread that is currently waiting on the wait condition;
(iii) puts at the end of the delay queue;
(iv) depends on the signalling discipline. [3 marks]

(g) In Java which two of the following methods are defined in class Thread?

start(); terminate(); run(); and wait();

(i) start() and run()
(ii) start() and terminate()
(iii) wait() and terminate()
(iv) run() and wait() [3 marks]

(h) Consider the following Java code fragment:

```
public class Example implements Runnable{
   public void run(){
                  // some code
                     }
               }
```

which one of the following will create and start this thread?

(i) new Example().start();
(ii) new Thread(new Example()).start();
(iii) new Thread(Example()).start();
(iv) new Example(new Thread()).start(); [4 marks]

2.

(a) What is spinlock? Explain why spinlock is not appropriate for single-processor systems yet are often used in multiprocessor systems. [5 marks]

(b) Explain how to use Test-and-Set instructions to solve a critical section problem. Add necessary Test-and-Set statements to ensure mutual exclusion for critical sections for the following two processes P1 and P2:

Process P1:

```
init1
while(true) {
  crit1
  rem1
}
```

Process P2:

```
init2
while(true) {
  crit2
  rem2
}
```

where init1 and init2 are non-critical initialisations, crit1 and crit2 are critical sections and rem1 and rem2 are non-critical remainders of the programs. Explain whether your modified program also ensures Absence of Livelock, Absence of Unnecessary Delay, and Eventual Entry. [10 marks]

(c) Consider the mutual exclusion protocol for 2 processes shown below:

```
Process P1:                                   Process P2:

    init1                                         init2
   while(true) {                                 while(true) {
      flag1 :=true;                                 flag2 := true;

    while(flag2) {};                              while(flag1) {};
       crit1                                         crit2
    flag1 :=false;                                flag2 := false;
       rem1                                          rem2
    }                                             }
                        // Shared variables
                Boolean flag1 = false, flag2 = false;
```

where init1, init2, crit1, crit2, rem1 and rem2 have the same meaning as before. Does the protocol satisfy the properties of Mutual Exclusion, Absence of livelock, and Eventual Entry? If not, give an example trace that results in the property being violated and fix the algorithm using only standard instructions so that it satisfies all the properties. State whether it has any advantages over the protocol you have corrected in part (b). [10 marks]

3.

(a) Define the notion of a semaphore and explain the behaviour of the primitive operations provided on semaphores. [5 marks]

(b)

(i) Consider the following two processes that share a common variable X:

```
Process P1:                                   Process P2:
  //initialisation code                       //initialisation code
    int Y;                                        int Z;
1:    Y = 2*X;                                1:    Z = X+1;
2:    X = Y;                                  2:    X = Z;
    //other code                                  //other code
                        // Shared variable
                              int X =2;
```

Assume that the assignments are executed atomically. How many different values of X are possible after both processes finish executing? What are those values? [6 marks]

(ii) Suppose the processes are modified using a shared binary semaphore as follows:

```
Process P1:                                   Process P2:
  //initialisation code                       //initialisation code
    int Y;                                        int Z;
    wait(S);                                      wait(S);
1:  Y = 2*X;                                  1:   Z = X+1;
2:  X = Y;                                    2:   X = Z;
    signal(S);                                    signal(S);
    //other code                                  //other code
                        // Shared variable
                              int X =2;
```

S is set to 1 before either process begins execution. Now, how many different values of X are possible after both processes finish executing? What are those values? [6 marks]

(c) The barber shop has one barber, one barber chair, and N (=5) chairs for waiting customers, if any, to sit in. If there is no customer at present, the barber sits down in the barber chair and falls asleep. When a customer arrives, he has to wake up the sleeping barber. If additional customers arrive while the barber is cutting a customer's hair, they either sit down (if there is an empty chair) or leave the shop (if all chairs are full). Using semaphores write a program to coordinate the barber and the customers. [8 marks]

4.

(a) Define the notion of a monitor and explain briefly how mutual exclusion and condition synchronisation within a monitor are achieved. [6 marks]

(b) Explain three major differences between a semaphore wait (also known as P)/signal (also known as V) and a condition variable wait/signal. [6 marks]

(c) Define the Dining-Philosopher problem for concurrent systems considering five Philosophers. Using monitor, devise a deadlock-free solution to the Dining-Philosopher problem for a shared resource. [13 marks]

5.

(a) What is a Thread in Java? Briefly explain Thread lifecycle in Java. [8 marks]

(b) Explain two different ways of implementing Java threads and state why one has advantage over the other. [7 marks]

(c) What is interference problem? State whether interference problem occurs in the following Java code fragment. If you answered in the affirmative, explain with example why it occurs and how will you fix it? [10 marks]

```
public class BankAccount {
        int accountNumber;
        double accountBalance;
        //other code

public boolean deposit(double amount)  {
      double newAccountBalance;
       if( amount < 0.0) {
         return false; // cannot deposit a negative amount
         }
        else {
         newAccountBalance = accountBalance + amount;
         accountBalance = newAccountBalance;
         return true;
        }
}
```

6.

(a) When model checking is applicable for analysing a safety critical concurrent program, explain why it can be more effective than other conventional approaches, e.g., software testing. [8 marks]

(b) Consider the following two processes:

```
condition = poor;

Process P1:                          Process P2:

while(true) {                        while(true) {
    condition = moderate;                condition = rich;
    condition = rich;                }
}
```

where condition is variable of *enum* type that takes one of the three possible values {poor, moderate, rich}. Assume that the assignments are executed atomically. Draw the state transition system for the processes executing concurrently. [8 marks]

(c) Express the following properties in CTL and state whether they hold in the initial state of the transition system from part (b).

(i) The condition is always poor, moderate or reach. [4 marks]

(ii) At every reachable state there is a path to a state where condition is rich. [5 marks]