2017 EXAMINATIONS

Part II

COMPUTING AND COMMUNICATIONS [2 Hours]

SCC.211 Operating Systems

Candidates are asked to answer THREE questions from FOUR; each question is worth a total of 20 marks.

Use a separate answer book for each question.

An appendix for question 3 *is included at the end of the paper*

Question 1

1.a

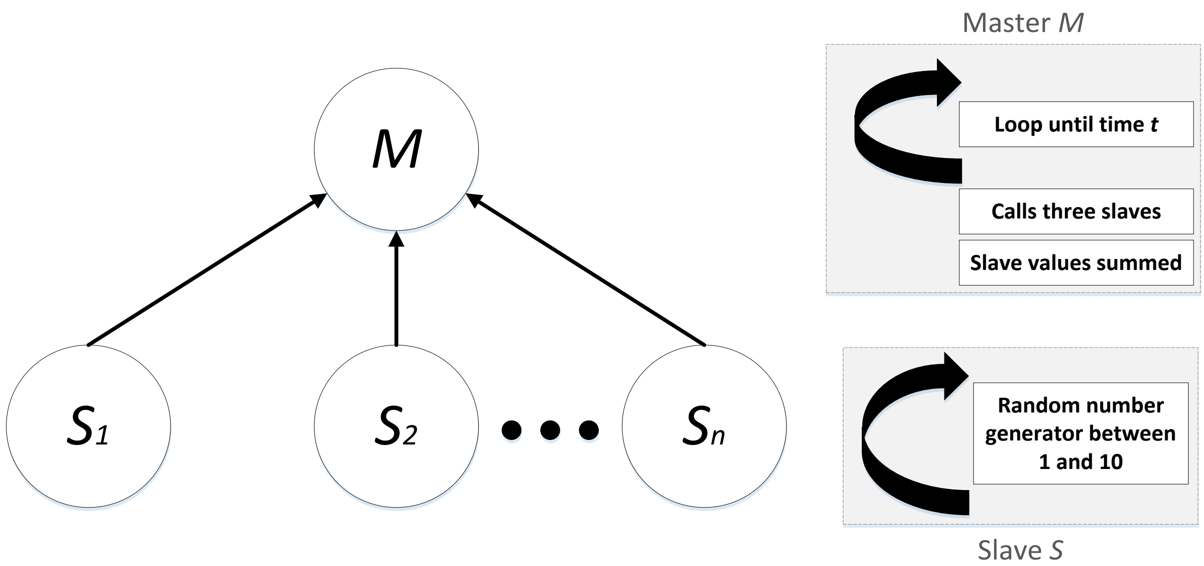
i. Define the distinction between concurrency and parallelism, and describe an example of a program which is concurrent but *not* parallel. **[3 marks]**

ii. Provide *two* motivations for concurrency **[2 marks]**

1.b

i. Provide *three* reasons why concurrent programmers are discouraged from relying on the scheduler to handle concurrency. **[3 marks]**

ii. Define the two main aspects of concurrency management required to achieve synchronization, and provide examples for each within the context of computing. **[4 marks]**



1.c The above diagram shows a scenario where Master thread *M* creates *n* slave threads S_n to perform computation. This computation is a random number generator, returning values between 1 and 10, residing within a loop. *M* also sits within a loop, and once every *t* seconds will unblock and call three slave threads at random to return their current random value generated. Values returned by the three slave threads are then summed by the Master.

i. In this scenario explain what are the potential risks with respect to synchronization. **[2 marks]**

Question 1 continues on next page…

Question 1 continued…

ii. Write an implementation for this scenario: You are free to use either Java-based or C-based pseudo-code and any concurrent programming primitives as you see fit. A precise description of the semantics of the concurrent primitives should be provided, which argues how they ensure non-determinism. Code should be as precise and complete as possible.

[6 marks]

Total 20 marks

Question 2

2.a

i. Define what is a kernel level thread and a user level thread. **[2 marks]**

ii. Provide *two* advantages of user threads over kernel threads. **[2 marks]**

iii. Provide *one* disadvantage of user threads compared to kernel threads. **[1 mark]**

2.b The following snippet of code provides multiple autonomous threads the ability to upload and download data into a database connected via a network. Locks are used to restrict access to the network and database from other threads during upload and download.

```
Object networkLock  = new Object();
Object databaseLock = new Object();

public void SysDownload( ) {
      lock(networkLock);
      lock(databaseLock);

            downloadData( );

      unlock(networkLock);
      unlock(databaseLock);
}

public void sysUpload( ) {
      lock(databaseLock);
      lock (networkLock);

            uploadData( );

      unlock(databaseLock);
      unlock(networkLock);
}
```

i. Explain the problems with this code, and why do they occur. **[3 marks]**

ii. Rewrite the above code snippet to provide a thread-safe synchronized solution. **[2 marks]**

Question 2 continues on next page…

Question 2 continued…

2.c The code in 2.b.ii has now been redeveloped so that the system cannot upload data unless it has first been downloaded and modified. The data modification is performed using a modify( ) method. It is possible for the system to contain up to five pending download requests, with subsequent uploads resulting in data removal from the database.

i. In this scenario semaphores represent one means to achieve synchronization; explain what the methods wait( ) and signal( ) represent. **[1 mark]**

ii. Provide a pseudo-code implementation for this scenario, using semaphores to achieve synchronization. Semaphore objects should be represented with syntax similar to:

```
Semaphore s = new Semaphore(x);
```

Where *x* indicates the semaphore counter value. The semaphore method and its semantics should also be included, which argues how it ensures non-determinism. Your code should be as precise and complete as possible. **[7 marks]**

iii. Discuss whether blocking or spin locks would be more effective in the context of the described scenario, and explain why.

[2 marks]

Total 20 marks

Question 3

3.a Two alternative implementations of a function forming part of a memory allocation scheme are provided in an appendix at the end of this paper.

i. Given the code for Algorithm A and Algorithm B in the appendix; explaining your reasoning clearly identify the memory allocation scheme each is implementing.

[4 marks]

ii. If the conditional in Algorithm A is changed to the following, which memory allocation scheme would be implemented?

```
if ( free && ( foundBlock == NULL ) && ( size >= requestSize )) {
```

[2 marks]

3.b A Round-Robin scheduler is presented with the following processes

Process	1	2	3	4	5
Arrival Time (ms)	0	5	25	30	35
Burst Time	20	30	10	25	5

i. Explain the terms:
   - Burst Time
   - Average Waiting Time
   - Average Turnaround Time
   - Scheduling Quantum

[4 marks]

ii. Showing all working and being careful to justify your answer in terms of average and total waiting and turnaround times, would it be better to have a scheduling quantum of 10 or 15ms for this scenario?

[10 marks]

Total 20 marks

Question 4

4.a Modern multi-tasking operating systems must manage possibly many different processes and both fairly and safely share resources between them.

i. Outline what we mean by a *cooperative scheduler* and a *pre-emptive scheduler* being careful to highlight the differences in terms of fairness and efficiency, and explain when and how each type of scheduler is invoked.

[8 marks]

ii. What do we mean by a *Deferred Procedure Call* (DPC), what purpose do they serve, and why this type of mechanism is common in a modern operating system.

[3 marks]

4.b To test the efficiency of a resource allocation scheme you look at frame allocation over a short execution period. The current configuration allocates *four frames* to process under a *First In First Out* (FIFO) scheme and the process experiences *ten page faults*.

Given the page reference string { 6, 3, 4, 5, 6, 3, 8, 6, 3, 4, 5, 8 }, would there be any advantage in restricting the number of frames to *three* under either a FIFO or *Least Recently Used* (LRU) allocation scheme? You should assume that no pages are loaded at the start and, must show all working and justify your answer.

[9 marks]

Total 20 marks

Appendix: *Code for question 3*

Algorithm A

```
/*
 *   Algorithm A
 *
 *   Find block for request of size requestSize
 *   Returns pointer to most appropriate block or NULL if no suitable block found
 */
FreeBlock *
getBlock ( uint64_t requestSize ) {


   /*
    *   Record most appropriate block found so far
    *   and its size
    */
   FreeBlock * foundBlock = NULL;
   uint64_t    foundSize  = MAXBLOCKSIZE;

   /*
    *   Iterate through list of free memory blocks (FreeList)
    */
   FreeBlock * ptr = FreeList;

   while ( ptr != NULL ) {
      uint64_t size = ptr -> blockSize;

      if (( size < foundSize ) && ( size >= requestSize )) {
         foundBlock = ptr;     // Keep a record of the block we've just found
         foundSize  = size;    // ...and its size
      }

      ptr = ptr -> next;       // Move to next block in FreeList
   }

   return foundBlock;
}
```

Algorithm B appears on next page...

Code for question 3 continued, Algorithm B

```
/*
 *   Algorithm B
 *
 *   Find block for request of size requestSize
 *   Returns pointer to most appropriate block or NULL if no suitable block found
 */
FreeBlock *
getBlock ( uint64_t requestSize ) {

   /*
    *   Record most appropriate block found so far
    *   and its size
    */
   FreeBlock * foundBlock = NULL;
   uint64_t    foundSize  = 0;

   /*
    *   Iterate through list of free memory blocks (FreeList)
    */
   FreeBlock * ptr = FreeList;

   while ( ptr != NULL ) {
      uint64_t size = ptr -> blockSize;

      if (( size > foundSize ) && ( size >= requestSize )) {
         foundBlock = ptr;      // Keep a record of the block we've just found
         foundSize  = size;     // ...and its size
      }

      ptr = ptr -> next;        // Move to next block in FreeList
   }

   return foundBlock;
}
```

--- End of Paper ---