2022 EXAMINATIONS

Lancaster University

Part II

COMPUTING AND COMMUNICATIONS – On-line Assessment [120 Minutes]

SCC.211 Operating Systems

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

Question 1

1a. Consider two threads with IDs 0 and 1, each with a critical section in which they access a shared resource. Assume they are using Peterson's algorithm for mutual exclusion (reproduced below from the lecture notes.)

```
//Peterson's algorithm
int tiebreak = 0;
bool[] interested = {FALSE, FALSE};

void get_lock() {
      int self = thread_getid();
      int other = 1-self;
      interested[self] = TRUE;
      tiebreak = other;
      while(interested[other] && tiebreak == other) ;
}

void release_lock() {
      int self = thread_getid()
      interested[self] = FALSE;
}
```

i. The following is an interleaved execution of the threads, where the prefix 0: and 1: indicate whether the instruction is from thread 0 or thread 1.

```
1: interested[1] = TRUE
0: interested[0] = TRUE
1: tiebreak = 0
0: tiebreak = 1
X: Critical Section
```

After the first four instructions, one of the two threads is able to enter its critical section. Should X be 0 or 1?

ii. Explain in at most two lines why Peterson's algorithm requires atomic memory reads and writes.

iii. Can Peterson's algorithm result in deadlock?

[6 marks]

[Please turn over]

Question 1 continues on next page…

1b. Suppose you have a program with two concurrent threads T1 and T2 that execute instructions for acquiring and releasing three locks (lock1, lock2, and lock3) in the order shown below

T1	T2
lock1.acquire lock1.release lock2.acquire lock3.acquire lock2.release lock3.release	lock1.acquire lock3.acquire lock2.acquire lock1.release lock3.release lock2.release

Show a shortest interleaved execution involving T1 and T2 that results in a deadlock.

Hint: To show an interleaved execution, show a single sequence of instructions, showing clearly which thread each instruction is from. See the interleaved execution in Question 1a for an example.

Hint: `Shortest' means there exists no interleaved execution with fewer instructions that results in a deadlock.

[6 marks]

1c. In less than four lines, give two motivations for supporting concurrency in software.

[4 marks]

1d. In less than four lines, explain with reference to the Dining Philosophers problem, why deadlock implies starvation but starvation does not imply deadlock.

[4 marks]

[Total 20 marks]

Question 2

2a. For each of the below statements about BSPL, state whether it is true or false and justify each answer in one sentence.

i. An idea behind BSPL is that all the shared information an agent needs to play a role in a protocol is passed explicitly through message parameters.

ii. BSPL does not assume ordered delivery from the communication infrastructure.

iii. A simple BSPL protocol such as *Hello* does not need a key.

iv. A parameter can have an **in** adornment in more than one message in a protocol.

v. An agent that already knows the binding of a parameter is forbidden from sending a message with an **out** adornment of the same parameter.

[10 marks]

2b. Explain what guarantees a communication service provides when we say that it guarantees *noncreative*, *reliable*, and *ordered* communication.

[6 marks]

2c. Consider the protocol Purchase given below.

```
Purchase {
   role B, S
   parameter out ID key, out item, out price

   B->S: want[out ID key, out item]
   B->S: willpay[in ID key, out price]
   S->B: offer[in ID key, out price]
}
```

Which of the statements below are true about Purchase?

i. Purchase is safe.
ii. Purchase is live.
iii. Eliminating the `willpay` message from the protocol will make it safe.
iv. Eliminating the `offer` message from the protocol will make it safe.

[4 marks]

[Total 20 marks]

[Please turn over]

Question 3

3.a An initially empty, and newly initialised, FAT filesystem receives a number of block writes for two new files F1 and F2. The first request is for F1, followed by F2, and requests are then served alternately for F1, F2, F1, … You should assume that the first FAT entry is index 1.

The requests are:

- F1: 2 blocks, 2 blocks, 3 blocks
- F2: 1 block, 4 blocks

Draw a diagram that clearly illustrates the final FAT state, including details of any additional information required for accessing the files.

[5 marks]

3.b

i. Illustrate **with two clear diagrams** how the fourteenth block of a file would be accessed in a FAT based system with each entry referencing a single block, **and** a classic Unix filesystem using combined allocation with twelve direct pointers.

ii. How many accesses would be required to access the data from the fourteenth block of the file in each scheme?

iii. **Justifying your answer** with clear reference to your diagrams, which would be fastest? …and why?

You should assume that for:

- FAT – the system already has a reference to the first entry for the file in the FAT
- Unix – the system already has a reference to the inode for the file.

[6 marks]

3.c 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

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?

[9 marks]

[Total 20 marks]

[Please turn over]

Question 4

4.a An operating system running on a 32 bit processor with a split/ two-level page table divides addresses into 10, 10, and 12 bits. Given this configuration:

Two processes, each exactly 8KiB, are loaded into memory, which is initially empty. The entirety of the first process is loaded, followed by all of the second.

Once both processes are loaded into memory, the first process then allocates 2KiB of memory on the heap using sbrk( ), and then the second process does the same.

i. Assuming there are no other processes in the system, there is no TLB, and ignoring any memory or page table entries that might be used by the kernel or other system elements, and the page tables themselves, **draw a diagram** that represents the data structures necessary to hold the page table mappings. **Your diagram must show**:

a) the memory areas occupied by the processes

b) how the data structures are linked and reference the memory areas

c) the number of free entries in each table you describe

[6 marks]

ii. **Justifying your answers** with specific reference to your diagram and the above values:

a) What is the maximum addressable memory within a process in this system?

b) How much memory would be needed to hold the processes and the page table data structures you have identified, assuming the processes request no memory beyond that outlined above?

c) At what point would such a system begin to experience fragmentation, and what form of fragmentation would this be?

[6 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 processes under a *First In, First Out* (FIFO) scheme and the processes experiences ten page faults.

Given the page reference string { 7, 4, 2, 5, 7, 4, 6, 7, 4, 2, 5, 6 }, 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**.

[8 marks]

[Total 20 marks]

--- End of Paper ---