Carleton UNIVERSITY

Sample Solution to Final
EXAMINATION
Winter 2018

DURATION: 3 HOURS **No. Of Students: 47**

Department Name & Course Number: Systems and Computer Engineering SYSC 4001B

Course Instructor: Thomas Kunz

AUTHORIZED MEMORANDA:

William Stallings, *Operating Systems: Internals and Design Principles*, 9th edition, Pearson 2018, ISBN-9780134670959 (as physical book, no ebook) or earlier versions of that same book.

Students MUST count the number of pages in this examination question paper before beginning to write, and report any discrepancy to a proctor. This question paper has 7 pages + cover page = 8 pages in all.

This examination question paper MAY NOT be taken from the examination room.

In addition to this question paper, students require: an examination booklet: NO
Scantron Sheet: NO

Name: ____

Student Number: ______________________________________

Question 1: ______ **/10**

Question 2: ______ **/10**

Question 3: ______ **/10**

Question 4: ______ **/10**

Question 5: ______ **/10**

Question 6: ______ **/10**

Total: ________**/60**

Exam questions will not be explained, and no hints will be given. If you think that something is unclear or ambiguous, make a reasonable assumption (one that does not contradict the question), write it at the start of the solution, and answer the question. Do not ask questions unless you believe you have found a mistake in the exam paper. If there is a mistake, the correction will be announced to the entire class. If there is no mistake, this will be confirmed, but no additional explanation of the question will be provided.

Question 1: Processes and Threats (10 marks)

1. Including the initial parent process, how many processes are created by the program below?

```
#include <stdio.h>
#include <unistd.h>
int main() {
      /* for a child process */
      fork();
      /* fork another child process */
      fork();
      /* and fork another */
      fork();
}
```

   ### Answer (4 marks):
   There are 8 processes created.

2. Provide three programming examples in which multithreading provides better performance than a single-threaded solution.

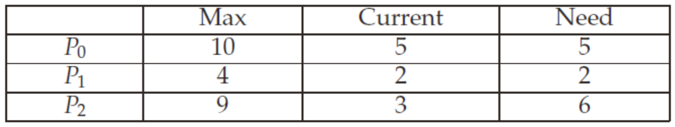
   ### Answer (4 marks):
   a. A Web server that services each request in a separate thread.
   b. A parallelized application such as matrix multiplication where different parts of the matrix may be worked on in parallel.
   c. An interactive GUI program such as a debugger where a thread is used to monitor user input, another thread represents the running application, and a third thread monitors performance.

3. What resources are used when a thread is created? How do they differ from those used when a process is created?

   ### Answer (2 marks):
   Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation. Creating a process requires allocating a process control block (PCB), a rather large data structure. The PCB includes a memory map, list of open files, and environment variables. Allocating and managing the memory map is typically the most time-consuming activity. Creating either a user or kernel thread involves allocating a small data structure to hold a register set, stack, and priority.

Question 2: Concurrency: Deadlocks (10 marks)

Assume a computer system with only one type of resources and three processes. The following table shows the current system state according to the Banker's algorithm:



	Max	Current	Need
P_0	10	5	5
P_1	4	2	2
P_2	9	3	6

Furthermore, assume that two more units of the resource are available.

1. Explain why the system is in an unsafe state.

   ### Answer (4 marks):
   Currently there are two resources available. This system is in an unsafe state as process P1 could complete, thereby freeing a total of four resources. But we cannot guarantee that processes P0 and P2 can complete (each requesting potentially more than 4 units of the resource).

2. Assume that, ultimately, each process will claim their max allocation. As the system is in an unsafe state, does that necessarily mean that a deadlock will occur? Explain your answer.

   ### Answer (6 marks):
   The Banker's algorithm assume a worst-case scenario, where processes end up claiming their max allocations without releasing any resources. But it is possible that a process may release resources before requesting any further. For example, process P2 could release a resource, thereby increasing the total number of resources to five. This allows process P0 to complete, which would free a total of nine resources, thereby allowing process P2 to complete as well.

Question 3. Memory Management (10 marks)

1. Consider the following page reference string:

   ```
   1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.
   ```

   How many page faults would occur for the following replacement algorithms, assuming one, four, or seven frames? Complete the table below. Remember all frames are initially empty, so your first unique pages will all cause one fault each

   - LRU replacement
   - FIFO replacement
   - Optimal replacement

   Answer (5 marks):

	LRU	FIFO	Optimal
One Frame	20	20	20
Four Frames	10	14	8
Seven Frames	7	7	7

2. Some page replacement algorithms have a strange property referred to as Belady's anomaly: if you have more frames, the number of page faults will actually increase. Take FIFO and the following page reference string:

   ```
   1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
   ```

   If the OS allocates 3 frames to the process, the number of page faults (including the initial three to fill the empty frames) is 9. If the OS allocates 4 frames, the number of page faults rises to 10. This is not, in general the expected behavior: as discussed in class, all else being equal, we expect the number of page faults to decrease as more frames are allocated to a process.

   You have devised a new page-replacement algorithm that you think may be optimal. In some contorted test cases, Belady's anomaly occurs. Is the new algorithm optimal? Explain your answer.

   Answer (5 marks):

   No. An optimal algorithm will not suffer from Belady's anomaly because —by definition—an optimal algorithm replaces the page that will not be used for the longest time. Belady's anomaly occurs when a page replacement algorithm evicts a page that will be needed in the immediate future. An optimal algorithm would not have selected such a page.

Question 4. CPU Scheduling (10 marks)

Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

```
    Process            Burst Time         Priority
       P1                  10                 3
       P2                   1                 1
       P3                   2                 3
       P4                   1                 4
       P5                   5                 2
```

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

a. Draw four charts (similar to Figure 9.5 in the textbook) that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SPN (also called SJF or Shortest Job First), non-preemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1). These figures (or similar representations) are also called Gantt charts.

b. What is the turnaround time of each process for each of the scheduling algorithms in part a? Complete the table below.

Turnaround Time	FCFS	SPN	Priority	RR
P1				
P2				
P3				
P4				
P5				

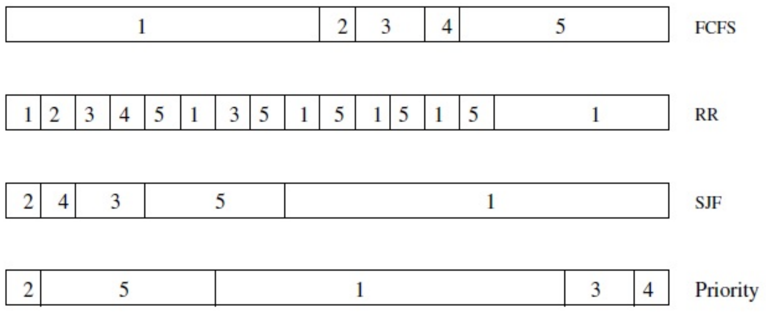
c. What is the waiting time of each process for each of these scheduling algorithms? Complete the table below.

Waiting Time	FCFS	SPN	Priority	RR
P1				
P2				
P3				
P4				
P5				

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Answers:

a. The four Gantt charts are



b. Turnaround time

	FCFS	RR	SJF	Priority
P_1	10	19	19	16
P_2	11	2	1	1
P_3	13	7	4	18
P_4	14	4	2	19
P_5	19	14	9	6

c. Waiting time (turnaround time minus burst time)

	FCFS	RR	SJF	Priority
P_1	0	9	9	6
P_2	10	1	0	0
P_3	11	5	2	16
P_4	13	3	1	18
P_5	14	9	4	1

d. Shortest Job First

Question 5. RAID (10 marks)

1. Consider a RAID Level 5 organization comprising five disks, with the parity for sets of four blocks on four disks stored on the fifth disk. Further, assume that the OS software minimizes the number of required disk I/O operations for each operation. How many blocks have to be read and are written to disk in order to perform the following?
   - A write of one block of data
   - A write of seven continuous blocks of data, starting at a four-block boundary.

   Explain your answer/reasoning

   ### Answer (5 marks):
   - A write of one block of data requires the following: read of the parity block, read of the old data stored in the target block, computation of the new parity based on the differences between the new and old contents of the target block, and write of the parity block and the target block, *for a total of 2 reads and 2 writes*.
   - A write of seven contiguous blocks of data could be performed by writing the seven contiguous blocks, writing the parity block of the first four blocks, reading the eight block, computing the parity for the next set of four blocks and writing the corresponding parity block onto disk. The *total is 1 read and 9 writes*.

2. Compare the performance of write operations achieved by a RAID Level 5 organization with that achieved by a RAID Level 1 organization. Which arrangement allows you to write individual disk blocks faster?

   ### Answer (3 marks):
   RAID Level 1 organization can perform writes by simply issuing the writes to mirrored data concurrently. RAID Level 5, on the other hand, would require the old contents of the parity block to be read before it is updated based on the new contents of the target block. This results in more overhead for the write operations on a RAID Level 5 system.

3. Assume that you have a mixed configuration comprising disks organized as RAID Level 1 and as RAID Level 5 disks. Assume that the system has flexibility in deciding which disk organization to use for storing a particular file. Which files should be stored in the RAID Level 1 disks and which in the RAID Level 5 disks in order to optimize performance?

   ### Answer (2 marks):
   Frequently updated data need to be stored on RAID Level 1 disks while data that is more frequently read as opposed to being written should be stored in RAID Level 5 disks.

Question 6. File Systems (10 marks)

1. Contrast the performance of the three techniques for allocating disk blocks (contiguous, linked, and indexed) for both sequential and random file access.

   ### Answer (6 marks):

   - Contiguous Sequential: Works very well as the file is stored contiguously. Sequential access simply involves traversing the contiguous disk blocks.
   - Contiguous Random: Works very well as you can easily determine the disk block containing the position you wish to seek to.
   - Linked Sequential: Satisfactory as you are simply following the links from one block to the next.
   - Linked Random: Poor as it may require following the links to several disk blocks until you arrive at the intended seek point of the file.
   - Indexed Sequential: Works well as sequential access simply involves sequentially accessing each index.
   - Indexed Random: Works well as it is easy to determine the index associated with the disk block containing the position you wish to seek to

2. What are the advantages of the variation of linked allocation that uses a FAT to chain together the blocks of a file? As discussed in class/the textbook, the FAT stores an entry for the first block, and the blocks are linked together via links stored with each disk block. The FAT as used in DOS and early versions of Windows, for example, contains entries for each block. Each entry contains either the number of the next block in the file, or else a marker indicating end of file, so no links need to be stored with the data blocks on disk: the OS can traverse the FAT to link together all disk blocks (which then still need to be retrieved via disk I/O).

   ### Answer (4 marks):

   The advantage is that while accessing a block that is stored at the middle of a file, its location can be determined by chasing the pointers stored in the FAT as opposed to accessing all of the individual blocks of the file in a sequential manner to find the pointer to the target block. Typically, most of the FAT can be cached in memory and therefore the pointers can be determined with just memory accesses instead of having to access the disk blocks.