A3

Below find three sections with homework questions. Submit your solutions to CMS in a separate PDF file called ``A3.pdf''. Please mark each answer clearly with the question number. For Q3(b) and Q3(c) you have to include tables in your answer.

Note that there are also synchronization programming problems to be done, described in a separate handout.

Q1. May the Odds Be Ever in Your Favor

In *Lottery Scheduling*, each process is assigned a disjoint subset of lottery tickets, and the scheduler selects a random single lottery ticket in order to select the next ready process to be run. Unlike with a real lottery, a specific lottery ticket might be selected in multiple rounds or in no round.

Giving more lottery tickets to a process increases the chances that process will be the next selected to next run by the scheduler. The resulting scheme is quite versatile. For example, processes can be given different priorities by allocating to them different fractions of the total number of tickets.

Assume that processes run forever (i.e., they are always runnable) and that processes do not communicate or synchronize with each other. In addition, suppose N_p lottery tickets that are assigned to each process p (for $0 <= p < T$, with T being the total number of processes).

a) Give a formula for the probability that a particular process p will next be selected to run by a scheduler that has access to an unbiased source of random numbers.

b) The scheduler runs every clock tick, which happens 100 times per second. What is the minimum and the maximum amount of time that a process might be delayed before it has an opportunity to execute?

a) Now suppose processes do communicate and synchronize with one another. One tricky aspect of implementing a scheduler that supports priorities is coping with *priority inversion (*where if a high priority process *p1* waiting on a lock held by another lower-priority process *p2*, then the priority of *p1* is effectively limited by the priority of *p2*). What rule could a Lottery Scheduling implementation adopt to mitigate this problem?

Q2. DisksRus

The Hardware R&D Group at DisksRus is having trouble developing a faster disk drive. Management is hoping for a miracle from a software update.

(a) The current product has the following specifications:

- it employs a single platter with data stored on only one surface
- There are 200 tracks on the surface
- each track stores 500 blocks of 512 bytes
- platter rotation speed is 3600 RPM
- seek time is 1 msec for each track that must be traversed

(i) What is the storage capacity (in bytes) of the disk?

(ii) What is the longest delay before a block transfer for a read or write request will start?

(iii) What is the maximum rate that data can be retrieved from the disk?

(b) An important DisksRus customer is threatening to switch to buying a product offered by the competition. The competitor's disk has 2x faster read-times, 2x faster write-times, but 1/4 the storage capacity. This customer's application rarely performs writes, so the delay it takes for a disk write to complete will not play a role in the customer's decision to switch. Read-times are important, though.

Management is asking you whether there is some way to change the disk driver so that a single DisksRus disk drive would exhibit performance that will keep the customer. Assume that I/O requests are to blocks that are uniformly and randomly distributed.

(iv) Describe your proposed solution, using 2 or 3 sentences.

(v) What is the storage capacity in bytes of the scheme you propose?

(vi) What is the longest delay before a block transfer for a read request will start and how does that compare to the answer to (a ii)?

(vii) What is the longest delay before a block transfer for a write request will start, and how does that compare to the answer to (a ii)?

(viii) What is the maximum transfer rate that data can be retrieved from the disk when it is accessed using your new driver?

Q3. Oldie but Goodie

The **PDP11** was a series of computers sold by Digital Equipment Corp. (DEC) from 1970 and into the nineties. A PDP11 computer has a 16-bit virtual address space, where each address identifies a byte, for a total of 64 Kbytes. A page is 2^{13} bytes = 8 Kbytes, and thus the virtual address space of a process consisted of 8 pages. A page table entry (PTE) had a 9-bit frame (= physical page) number, a Valid bit, and a Writable bit.

a) What is the maximum physical memory (in Kbytes) in a PDP11? (A Kbyte is 1024 bytes.)

b) Consider the following page table of a process:

Page	Valid	Frame	Writable
0	yes	0x003	no
1	yes	0x001	no
2	yes	0x008	yes
3	no	N/A	N/A

Page	Valid	Frame	Writable
4	no	N/A	N/A
5	no	N/A	N/A
6	no	N/A	N/A
7	yes	0x004	yes

Fill in the following table:

Virtual Address	Valid (yes, no)	Physical Address (if valid) in hexadecimals	Writable (yes, no)
0x1234			
0x4321			
0x8888			

c) The Bogux O/S running on the PDP11 uses "Local Replacement", meaning that it assigns a certain number of physical frames to each process. As a result, two processes never contend for the same frame. However, if a lot of processes are running, the number of frames per process may well be fewer than 8. Assume a situation in which each process has three frames. Suppose the page reference string of some process is

0, 7, 2, 0, 7, 1, 0, 3, 1, 2

Initially no pages are mapped to physical frames. Now consider the state of the process's page table after the first 7 references (i.e., after page accesses 0 7 2 0 7 1 0). Which (up to three) pages are mapped at this time assuming one of the following page replacement schemes, and how many page faults have occurred then. Also show in the last column how many page faults occur in total after all 10 references?

Scheme	all page numbers of mapped pages **after 7 references** (3 max.)	#page faults **after 7 references**	#page faults total (**after 10 references**)
First In First Out (by way of example)	0 1 2	5	7
LRU (Least Recently Used)			
OPT (Belady)			