DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
CONCORDIA UNIVERSITY
COMP 346: Operating Systems
Fall 2019
Theory Assignment 3
Due date: Friday November 29th before 11:59 pm

Total marks: 90

Q.1 [10 marks] Consider the following preemptive priority-scheduling algorithm based on dynamically changing priorities. Larger priority numbers indicate higher priority (e.g., priority 3 is higher priority than priority 1; priority 0 is higher priority than priority -1, etc.). When a process is waiting for the CPU in the ready queue, its priority changes at the rate of α per unit of time; when it is running, its priority changes at the rate of β per unit of time. All processes are assigned priority 0 when they enter the ready queue. Answer the following questions:

a) What is the scheduling algorithm that results when $\beta > \alpha > 0$? Explain.
b) What is the scheduling algorithm that results when $\alpha < \beta < 0$? Explain.

Q.2 [10 marks] An enhanced Round Robin CPU scheduling algorithm works as follows: from the set of ready processes, pick the one to run which used the smallest fraction of the quantum during the last time it ran. A newly-arrived process is assumed to have used half its quantum. Also assume a constant steady stream of newly arriving processes. Answer the following questions:

a) Will the algorithm cause starvation? Explain your answer.
b) If your answer to part a) is yes, suggest an enhancement to the algorithm to resolve starvation.

Q.3 [10 marks] Consider the version of the dining philosopher's problem in which the chopsticks are placed in the centre of the table and any two of them can be used by a philosopher. Assume that requests for chopsticks are made one chopstick at a time. Describe a simple rule for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

Q.4 [10 marks] Consider Q.3 above. Assume now that each philosopher needs three chopsticks to eat. Resource requests are still issued one at a time. Describe some simple rules for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

Q.5 [10 marks] Consider a system consisting of m resources of the same type being shared by n processes. A process can request or release only one resource at a time. Show that the system is deadlock free if the following two conditions hold:

1. The maximum need of each process is between 1 resource and m resources.
2. The sum of all maximum needs is less than $m + n$.

Q.6 [10 marks] In a 64-bit computer system that uses pure paging with 16KB page size, if each page table entry is 4 bytes long then:

i. What is the maximum size of a page table?
ii. What is the maximum size of user-space main memory that can be supported by this scheme?
iii. If hierarchical paging is used then what is the total level of hierarchies required? Assume that 2-level of hierarchy corresponds to an outer page table and an inner page table. Show all relevant calculations.

iv. If the inverted paging scheme is used instead of paging, with a 16KB page/frame size, then what is the maximum size of the inverted page table considering the same memory size calculated in (ii) above? Assume that each entry of the inverted page table is 4 bytes long.

Q.7 [10 marks] Consider a demand-paged system where the page table for each process resides in main memory. In addition, there is a fast associative memory (also known as TLB which stands for Translation Look-aside Buffer) to speed up the translation process. Each single memory access takes 1 microsecond while each TLB access takes 0.2 microseconds. Assume that 2% of the page requests lead to page faults, while 98% are hits. On the average, page fault time is 20 milliseconds (includes everything: TLB/memory/disc access time and transfer, and any context switch overhead). Out of the 98% page hits, 80 % of the accesses are found in the TLB and the rest, 20%, are TLB misses. Calculate the effective memory access time for the system.

Q.8 [10 marks] Consider the two dimensional integer array A:

```
int A[][] = new int[100][100];
```

Assume that page size of the system = 200 bytes, and each integer occupies 1 byte. Also assume that A[0][0] is located at location 200 (i.e., page 1) in the paged memory system, and the array is stored in memory in row-major order (i.e., row by row). A small process that manipulates the array resides at page 0 (locations 0 to 199). Thus every instruction fetch is from page 0. The process is assigned 3 memory frames: frame 0 is already loaded with page 0 (i.e., process code), and the other two frames are initially empty.

a) Consider the following code fragment for initializing the array:

```
for (int i = 0; i < 100; i++)
        for (int j = 0; j < 100; j++)
            A[i][j] = 0;
```

If the LRU (Least Recently Used) page replacement scheme is used then what is the total number of page faults in executing the previous initialization code?

b) Now, instead, consider the following code fragment for initializing the array:

```
for (int j = 0; j < 100; j++)
      for (int i = 0; i < 100; i++)
          A[i][j] = 0;
```

If the LRU page replacement scheme is used then what is the total number of page faults in executing the above initialization code?

Q.9 [10 marks] Which of the following programming techniques and data structures (in a user-level program) are good for a demand-paged environment, and which are bad? Explain your answer.

i. Sequential search through a linked list
ii. Sequential search through an array
iii. Binary tree search
iv. Hashing with linear probing
v. Queue implemented using a circular array.