Department of Computer Science
University of Saskatchewan
CMPT 332
Midterm Exam

October 25, 2019	Name:_____
Number of Pages: 6	Signature:_____
Time: 50 minutes	Student Number: _____
Total: 50 marks.	NSID: _____

CAUTION - Candidates suspected of any of the following, or similar, dishonest practices shall be dismissed from the examination and shall be liable to disciplinary action.

1. Having at the place of writing any communication devices (including cellphones), any books, papers or memoranda, calculators, audio/visual players of any kind, or other memory aid devices.
2. Speaking or communicating with other candidates.
3. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	TOTAL
5	3	4	12	12	3	3	5	3	50

1. (5 marks) Assume you have 640 KB of main memory. Processes A, B, C, D, E and F with their associated memory requirements are presented to the medium-term scheduler for execution in the following order. How is memory allocated for the following sequence of operations for loading these 6 processes into memory using the **First Fit** algorithm? Show your answer by providing one diagram of RAM **after** all the allocation and deallocation has been completed.

Process	Size (in KB)
A	100
B	180
C	244
D	98
E	240
F	70

Operations to perform:

- Allocate A,
- Allocate B,
- Allocate C,
- Free B,
- Allocate E,
- Allocate D,
- Allocate F,
- Free C.

IF the operations cannot be completed in the given order, indicate which operations would have to be aborted. **Do not** complete an aborted operation at a later time.

2. (3 marks). Semaphore operations are just another example of the critical section problem (i.e. they access their shared data in a mutually exclusive way). Why is it OK to use a software (or hardware) solution (such as the ones presented in class) to make the semaphore operations atomic while the use of software solutions for general critical section problems is frowned upon? (a few sentences max).

3. (4 marks). Assume two processes i and j. Process i is listed below. Process j is identical, except with the i's and j's reversed. The variable *turn* and the array *flag[]* are shared. Assume that the *flag* array is initialized to false and that *turn* is initialized to either i or j.

```
PROCESS i:
while (1)
{
        flag[i] = true;
        turn = j;
        while (flag[j] && (turn == j)); /* busy wait */

        CRITICAL SECTION
        flag[i] = false;

        NON-CRITICAL SECTION
}
```

Does this algorithm ensure mutual exclusion? (yes or no). If not, give a brief counter example. If yes, does it solve all of the problems associated with concurrent access?

4. (12 marks) Process Scheduling. Consider the following job mix, including relative arrival times.

Job	Arrival Time	CPU Burst Time
A	0	10
B	5	7
C	6	8
D	4	7
E	5	4

Show the order of execution of these jobs using Round-Robin with a quantum of 4 and Shortest Remaining Time First. Use a time-line like in the interactive examples and/or class showing which jobs execute during which time slots. For RR, jobs are assumed to arrive in **alphabetical order** just **before** the time slot indicated and are placed at the **back** of the ready queue.

Indicate the avg. turnaround time (TAT) and the **specific CPU efficiency** for each algorithm on this set of tasks. For the *efficiency*, assume that each context switch takes **0.2 of a time unit**. For the *timeline*, assume that each context switch takes **0** time units. No calculator is necessary; leave the efficiency as a fraction.

RR

```
-------------------------------------------------------------------------------------
|    |    |
-------------------------------------------------------------------------------------
 0   4    8   etc...
```

TAT: Specific CPU EFFICIENCY for this run:

SRTF

```
-------------------------------------------------------------------------
|        |
-------------------------------------------------------------------------
 0
```

TAT: Specific CPU EFFICIENCY for this run:

5. (12 marks). Semaphores, Monitors, and Rendezvous IPC can all solve the same classes of problems, in that you can implement one facility with one of the others. Use Monitors to implement the Semaphore Operations P() and V(). Provide detailed **C-like pseudocode**. Identify the List Operations that you need (if you need any), but do not implement them. Assume that you have a monitor facility in your programming language, so that you do not need MonEnter and MonLeave. Assume also that the monitor contains the PCB for all the processes. Define those fields in the PCB which you think are necessary (in particular, condition variables). I've got you started. The implementation is short.

```
Monitor SEMAPHORES
{
/*shared  data and initialization */









/* Monitor procedures  */
int P(                                  int V(
{                                            {
```

```
}                                           }

void Initialize (
{
```

```
}
```

6. (3 marks) Briefly describe the activities the OS performs when executing a context switch.

7. (4 marks) Briefly outline one approach for each of deadlock detection and deadlock avoidance. What is one advantage of one of your suggestions over the other?

8. (4 marks) We studied the administrator model for rendezvous IPC and client-server applications. Now, assume that two administrators wish to communicate with each other. Describe a mechanism for this communication.

9. (3 marks) The priority increment for returning from a keyboard input is 2. The priority increment for returning from a disk input is 1. The highest MLF priority is 4. The maximum time allowable at level 4 is Q and doubles at each lower level. The **base priority** of process p is 1.

   - p runs for 3Q starting at level 3
   - p blocks on keyboard input
   - p continues running for 9Q
   - p blocks on disk input
   - p continues running for 4Q
   - p blocks on keyboard input
   - p continues running for 4Q

———————- THE END———————