Department of Computer Science
University of Saskatchewan
CMPT 332
Midterm Exam

October 27, 2017
Number of Pages: 6
Time: 50 minutes
Total: 50 marks.

Name:_____
Signature:_____
Student Number: _____
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 **Best 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 D,
- Allocate A,
- Allocate C,
- Free A,
- Allocate E,
- Allocate B,
- Allocate F,
- Free D.

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). What information (and in what order) must be saved at context switch time to adequately describe the state of a process? Be as complete and explicit as possible.

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	8	8
D	2	7
E	7	4

Show the order of execution of these jobs using Round-Robin with a quantum of 3 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 just **before** the time slot indicated and are placed at the **back** of the ready queue.

```
RR

-------------------------------------------------------------------------------------
|    |    |
-------------------------------------------------------------------------------------
 0   3    6   etc...

TAT:                                                    Specific CPU EFFICIENCY:
```

```
SRTF

-------------------------------------------------------------------------
|        |
-------------------------------------------------------------------------
 0

TAT:                                                    Specific CPU EFFICIENCY:
```

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.

5. (12 marks). You are writing an application using an operating system designed by a fool. The problem with the operating system is that the Sleep() call is not accessible to application programs, it is only available to special system processes (Sleep(N) suspends execution of the calling process for N seconds). The application you are writing requires this call, but cannot run as a system process. You therefore, in a fit of brilliance, decide to write a sleep server. The sleep server will accept sleep requests from clients and suspend the client's execution for the desired duration. This server (and its workers, if any) are system processes and therefore have access to the builtin Sleep() call.

   Notes:

   - you must write the stub routine MySleep(int seconds), the server and any workers
   - your server must be able to handle new requests while acting on previous ones
   - if you use the administrator model, you must NOT create a worker process every time MySleep() is called. You may only create one if your existing workers are all occupied.
   - assume pthreads-like IPC and process management primitives, and list primitives (if needed).
   - please use C-like pseudo-code

6. (4 marks) What are the differences and similarities between MonSignal(), MonNotify(), V(), and Reply()?

Similarity	Difference

7. (3 marks) What two advantages do threads have over multiple processes? What major disadvantage do they have?

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) List (in point form) 6 factors that must be considered in the scheduling algorithms that were covered in class (uniprocessor, multiprocessor, and realtime). There are more than 6.

———————- THE END———————