Department of Computer Science
University of Saskatchewan
CMPT 332
Midterm Exam

October 28, 2016	Name:_____
Number of Pages: 5	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, any books, papers or memoranda, calculators, audio or visual cassette players, 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	4	5	12	10	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 **Worst 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	180
B	100
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. (4 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. (5 marks). For the following Fork/Join code, draw the Process Flow Graph and give the Cobegin/Coend code (if possible) that corresponds to that particular code sequence.

```
     P1
     fork L2
     fork L3
     P5
     goto L8
L2:  P2
     fork L6
     P7
     goto L4
L3:  P3
L4:  join 2
     P4
     goto L8
L6: P6
L8:  join 3
     P8
```

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	6
D	2	7
E	7	5

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 .2 of a time unit. For the timeline, assume that each context switch takes 0 time. No calculator is necessary; leave the efficiency as a fraction.

5. (10 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 IPC primitives Send, Receive and Reply as defined in class to have zero-capacity semantics. 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, and that messages consist of a pointer to shared space and a buffer length. 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 IPC
{
/*shared  data and initialization */
```

```
/* Monitor procedures  */
char *Send(
{
```

```
}
```

```
char *Receive(
{
```

```
}
```

```
int Reply (
{
```

```
}
```

6. (3 marks) If a multithreaded process calls fork, a problem occurs if the child gets copies of all the parent's threads. Suppose that one of the original threads was waiting for keyboard input. Now two threads are waiting for keyboard input, one in each process. Does this problem (or a similar problem) ever occur in single-threaded processes? Why or why not?

7. (3 marks) What are three challenges for inter-machine communication (whether message passing or RPC) from the application's point of view? What is one solution for each challenge, if there is a solution?

Challenge	Solution

8. (5 marks) You've been asked to design and implement a communication protocol. The protocol will be used to support communication for a server your company uses. You ask your boss

   "Does the server in this communication protocol provide an idempotent or a non-idempotent service?"

   Your boss gives you a look like you're nuts and asks you what you are talking about. Please define the meaning of the word idempotent in this context, and explain why it is important for you to know this given the task at hand.

9. (3 marks) What is one activity of the CPU scheduler that would **only** be done for the Earliest Deadline First or Rate Monotonic Algorithms? Why is this important in these cases?

———————- THE END———————