Department of Computer Science
University of Saskatchewan
CMPT 332
Midterm Exam

October 31, 2014
Number of Pages: 5
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, 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	TOTAL
5	6	5	12	10	4	3	5	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	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. (6 marks). What are three various **categories** of information about processes that are stored for a multi-threaded process? Give an example piece of information of **each** category and **why** it belongs to that category.

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	8
C	4	6
D	2	4
E	7	5

Show the order of execution of these jobs using Round-Robin with a quantum of 2 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   2  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 S/R/R IPC to implement the Monitor primitives MonEnter, MonLeave, MonWait, and MonSignal as defined in class, and a MonitorServer thread. Provide detailed C-like pseudocode. Identify and use the List Operations that you need, but do not implement them. Assume that messages consist of a pointer to shared space and a buffer length. I've got you started.

```
int MonEnter(                  )          int MonLeave(                  )
{                                         {

}                                         }

int MonWait(                   )          int MonSignal(                  )
{                                         {

}                                         }

void MonServer (                            )
{

}
```

6. (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?

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) Kernel-level threads and user-level threads provide the same interface/functionality to the application programmer. Internally, however, they are much different. Describe these differences in regards to the **complexity of the implementation** and the **performance consequences**.

——————————- THE END——————————