Department of Computer Science
University of Saskatchewan
CMPT 332
Midterm Exam

October 26, 2022	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	5	12	12	3	3	4	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	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). 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	3	8
D	4	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.

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   3    6   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 IPC primitives Send, Receive and Reply as defined in class to have zero-capacity semantics (Blocking S/R/R). Provide detailed **C-like pseudocode**. Assume the following:

   - you have list operations should you need them.
   - you have a monitor facility in your programming language: you do not need MonEnter/MonLeave.
   - messages consist of a pointer to shared space and a buffer length.
   - the monitor contains the PCB for all the processes and fields can be accessed via `pid.fieldname`.

   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 */              /* PCB fields */









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









}                                            }

int Reply (
{




}
```

6. (3 marks) Briefly describe the similarities and the differences between a context switch and a system call.

Similarities	Differences

7. (3 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) List at least two differences and two similarities between P(sem)/MonWait()/Send()

9. (3 marks) Briefly explain how real-time and multi-processor scheduling algorithms differ from uniprocessor best-effort scheduling algorithms. What additional factors are important?

——————————- THE END——————————