BRAC UNIVERSITY
Department of Computer Science and Engineering
CSE321: Operating Systems

Final Exam **Fall 2021**
Duration: 1 Hour 40 minutes **Total Marks: 50**

---

1. a) Consider the following snapshot of a system. [6+3]

Allocation					
	A	B	C	D	E
T0	4	2	0	2	1
T1	2	1	5	0	2
T2	3	2	2	1	1
T3	1	3	5	1	1
T4	0	1	3	1	3

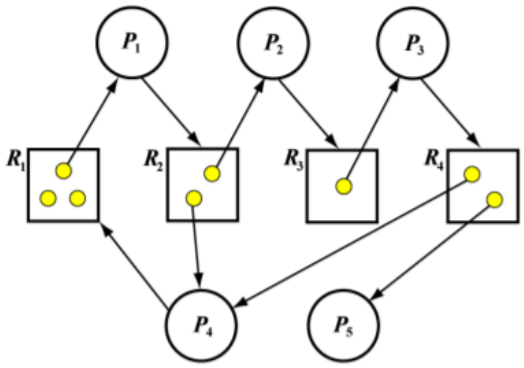
Max					
	A	B	C	D	E
T0	4	4	2	2	1
T1	3	1	5	2	2
T2	3	4	2	4	1
T3	2	3	5	3	1
T4	3	1	3	2	3

Available				
A	B	C	D	E
1	1	1	2	1

i) Using the banker's algorithm, determine whether or not the state is unsafe. If the state is safe, illustrate the order in which the processes may complete. Otherwise, illustrate why the state is unsafe. You must calculate the Need matrix.

ii) If a request from process T1 arrives for (0,4,2,0,2), can the request be granted immediately? Explain with proper calculations.

b) Consider the following resource allocation graph. [3+1]



i) Convert it to the matrix representation (i.e., Find matrix for Allocation, request, and Available).

ii) Is there a deadlock? If there is a deadlock, which processes are involved?

c) What is the hold and wait condition for deadlock? Explain why violation of this condition is sufficient to prove that there is no deadlock among the set of processes even if mutual exclusion and no-preemption conditions are valid? [1+1]

2. a) Given Static memory partitions shown in the following picture, Draw diagrams on how would each of the first-fit, best-fit, worst-fit algorithms place processes with the space requirement of the processes - P1 (146 KB), P2(425 KB), P3(240 KB), P4(89 KB) and P5(450 KB) (in order). Which algorithm makes the most effective use of memory? Answer with justification. [6+2]

Memory Partitions:	200 KB	250 KB	450 KB	160 KB	320 KB	150 KB	600 KB

b) Assume that in a certain situation of paged memory management system page size is 2KB, find the page number and offset for the following logical addresses of 32 bit or references (provide as a decimal number): [4]

i) 1085
ii) 30000
iii) 19467
iv) 15385

c) Describe the advantages of paging over segmentation. [3]

3. a) Remember this readers-writers synchronization problem from BUX and the reference book. [8]

```
do {
   wait(rw_mutex);
      . . .
   /* writing is performed */
      . . .
   signal(rw_mutex);
} while (true);
```

Figure 5.11 The structure of a writer process.

```
do {
   wait(mutex);
   read_count++;
   if (read_count == 1)
      wait(rw_mutex);
   signal(mutex);
      . . .
   /* reading is performed */
      . . .
   wait(mutex);
   read_count--;
   if (read_count == 0)
      signal(rw_mutex);
   signal(mutex);
} while (true);
```

Figure 5.12 The structure of a reader process.

This solution solves the critical section problem. However, there is a starvation issue here for the writer process. If there is a steady stream of reader processes in the ready queue, the writer process may starve and may not be able to write. We hope you can solve this starvation issue by applying one restriction in the code. The restriction is that if the writer was waiting for a reader, then that reader cannot read again before the writer completes writing. If more than one reader were reading when the writer was waiting, this restriction applies to all those readers. Update the reader and writer code to implement this restriction.

HINTS:

- You do not need any additional mutex for this but need to use both

mutexes in both programs.

- You need a flag variable shared among the writer and readers to indicate if the writer is waiting. Call it the **writer_waiting** flag.
- You need a private variable in the reader process, call it **i_kept_writer_waiting**, so that a reader can remember when starting its next iteration that the writer was waiting when it was reading in the previous iteration.

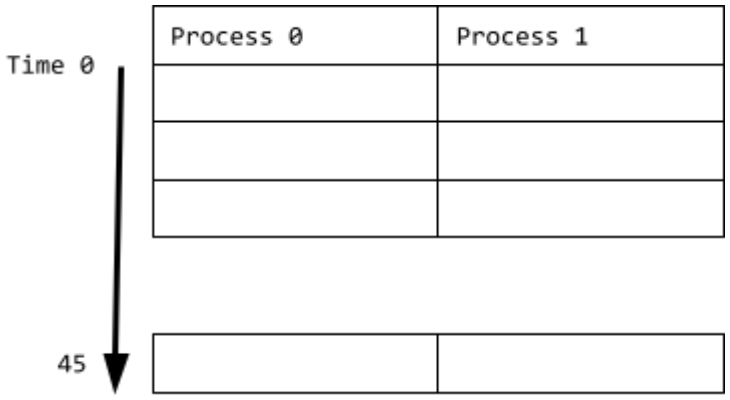
[6]

b) For Peterson's problem, the conditions below will apply.

- Each statement will take 3ms to complete.
- For process 0: i=0,j=1; and for process 1: i=1,j=0.
- Context switching will occur after every 9ms.
- In the critical section area, there are only 2 statements.
- The remaining section area contains only 1 statement.
- Information common to both processes:

```
turn=0;
flag[0]=FALSE;
flag[1]=FALSE;
```

Complete the following table **up to 45ms** in the timeline considering the above conditions and information. In the table, you will write the corresponding lines of code each process executes in that time slot.



The pseudocode for Peterson's solution is given below for any process P_i,

```
do {
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j);

        critical section

    flag[i] = false;

        remainder section

} while (true);
```

[3]

c) We all know the producer-consumer problem. Explain in general terms, how a solution to the producer-consumer problem can help to serve the guests in a

wedding ceremony problem. The food is brought to all the tables then the first batch of guests eat. After the guests complete eating, the tables are again filled with food so that the next batch of guests can start. Don’t forget to mention who is the producer, who is the consumer, and what is the buffer in this scenario. Do you need any change in the producer-consumer solution to support this scenario? If yes then explain the change.

[3]

d) Name the three conditions any solution for the critical section must meet. Give an example of a problem that can happen if any condition is violated for all three conditions.