United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Final Exam : Fall - 2018

Course: CSI 309 Operating System Concepts

Marks: 40, Time: 2 hours

Figures in the right-hand margin indicate full marks.

1.	a) Explain what will happen if we insert a program multiple times in a round-robin scheduling queue and why?	**3**
	b) What are the pros and cons if we set time quantum length in interactive systems too high or too low?	**3**
	c) For an interactive system, consider the table and so far scheduled Gantt chart below:	**4**

Process	Promised CPU time	Arrival time(s)
A	10%	0
B	35%	10
C	25%	40
D	20%	70

A(10s)	B(27s)	A(5s)	C(27s)	B(22s)	D(10s)	**?**

At a certain instance of time process D went to block state after running for 10s. If all other processes (A, B, C) are in ready state at that moment, which process will be scheduled next if the system follows guaranteed scheduling scheme? Show al necessary calculations.

2.	a) Does the busy waiting solution using the ***turn*** variable (strict alteration / Peterson's Solution) work when the two processes are running on a shared-memory multiprocessor, that is, two CPUs sharing a common memory?	**2**
	b) Suppose that we have a message-passing system using mailboxes. When sending to a full mailbox or trying to receive from an empty one, a process does not block. Instead, it gets an error code back. The process responds to the error code by just trying again, over and over, until it succeeds. Does this scheme lead to race conditions? If yes, how? If not, why?	**3**
	c) What will happen if two processes call enter_region function of Peterson's solution simultaneously?	**3**
	d) What is priority inversion problem in busy waiting?	**2**

3.	a) Define semaphore, counting semaphore and binary semaphore.	2
	b) Consider the given solution of well-known dining philosopher problem below:	2+2

```
#define N           5
#define LEFT        (i+N−1)%N
#define RIGHT       (i+1)%N
#define THINKING    0
#define HUNGRY      1
#define EATING      2
typedef int semaphore;
int state[N];
semaphore mutex = 1;
semaphore s[N];

void philosopher(int i)
{
    while (TRUE) {
        think( );
        take_forks(i);
        eat( );
        put_forks(i);
    }
}
```

```
void take_forks(int i)
{
    down(&mutex);
    state[i] = HUNGRY;
    test(i);
    up(&mutex);
    down(&s[i]);
}

void put_forks(i)
{
    down(&mutex);
    state[i] = THINKING;
    test(LEFT);
    test(RIGHT);
    up(&mutex);
}
```

```
void test(i)                    /* i: philosopher number, from 0 to N−1 */
{
    if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {
        state[i] = EATING;
        up(&s[i]);
    }
}
```

i) In the solution to the dining philosophers problem, why is the state variable set to HUNGRY in the procedure take-forks?

ii) Consider the procedure put_forks. Suppose that the variable state[i] was set to THINKING after the two calls to test, rather than before. How would this change affect the solution?

c) Using semaphore modify process A and B in such a way that overall execution sequence will be like: Authenticate → refresh → Logout → refresh → Authenticate → refresh → **4**

Process A	Process B

While(True){
Authenticate()
Logout()
}
	While(True){
	refresh()
	}

4.	a) State the 4 requirements of deadlock	**2**
	b) Consider the following scenario: ➢ Process *A* holds *R* and wants *S.* ➢ Process *B* holds nothing but wants *T.* ➢ Process C holds nothing but wants *S.* ➢ Process *D* holds *U* and wants *S* and *T.* ➢ Process *E* holds T and wants *V.* ➢ Process *F* holds *W* and wants *S.* ➢ Process *G* holds *V* and wants *U.* i) Draw R.A.G.(Resource Allocation Graph) from the given scenario. ii) Is there any deadlock in the above system? Mention reasons behind your answer. iii) State a sequence of process execution that will ensure that all processes finish executing properly(with minimum resource preemption).	**2+2 +2**
	c) What is the rationale behind using Ostrich Algorithm?	**2**

5.	a) For the resource allocation scenario given below, if process B asks for another instance of R4, find out whether the system will approve or deny the request according to Banker's Algorithm].	**3**
	b) Why linked-list disk block allocation using a table in memory is not a scalable solution?	**2**
	c) What are the drawbacks of hard linked file sharing with i-node? Explain with an example.	**2**
	d) Give a brief description of file system layout.	**3**

Resource Assigned (Question 5 a)

	R1	R2	R3	R4
A	0	2	1	0
B	3	0	1	1
C	0	4	5	1
D	1	1	1	1
E	0	1	0	0

Resource Still needed

	R1	R2	R3	R4
A	2	1	0	3
B	0	5	3	2
C	3	5	0	1
D	2	0	3	0
E	2	2	2	6

Here, available matrix is:[2 1 1 4]