BIRZEIT UNIVERSITY
Faculty of Engineering and Technology
Electrical and Computer Engineering Department

ENCS339: OPERATING SYSTEMS
Midterm Exam 2

Dr. Ahmad Alsadeh
May 3, 2015
Exam duration: **90** minutes

Name:_____ **Solution Key**_____ **ID**:_____

Question	Possible	Score	ABET Outcome
1	30		e
2	20		c
3	30		a
4	20		c
Total			

Question1 (30 points) (*ABET outcome e: Ability to identify, formulate and solve engineering problems.)*
Please fill the answer for the multiple choice question in the table

1	2	3	4	5	6	7	8	9	10
c	**b**	**b**	**c**	**b**	**b**	**b**	**b**	**b**	**a**
11	12	13	14	15	16	17	18	19	20
c	**c**	**b**	**a**	**b**	**d**	**b**	**b**	**b**	**c**

1. A race condition is:
a) When one process is trying to beat another to execute a region of code.
b) When a process cannot make progress because another one is blocking it.
c) When the outcome of processes is dependent on the exact order of execution among them.
d) A form of locking where processes coordinate for exclusive access to a critical section.

2. One example of a hardware solution to the critical section problem is:
a) Peterson's Algorithm
b) Test and Set
c) Compare and Shop
d) Compare and Pray

3. An instruction that executes atomically ____.
a) must consist of only one machine instruction
b) executes as a single, uninterruptible unit
c) cannot be used to solve the critical section problem
d) All of the above

4. In Peterson's solution, the ____ variable indicates if a process is ready to enter its critical section.
a) `turn`
b) `lock`
c) `flag[i]`
d) `turn[i]`

5. Which one is high-level language structure to solve the critical-region problem?
a) Test And Set Lock
b) Monitor
c) Shared memory
d) Semaphore

6. Starvation is the case when a thread:
a) Loops continuously until it runs out of memory.
b) Is never scheduled to run.
c) Can never acquire a lock on a critical section.
d) Cannot create a child process or thread

7. In scheduling the term *aging* involves
a) higher priority processes preventing low-priority processes from ever getting the CPU
b) gradually increasing the priority of a process so that a process will eventually execute
c) processes that are ready to run but stuck waiting indefinitely for the CPU
d) processes being stuck in ready queue so long that they are terminated

8. ____ is the amount of time a process has been waiting to enter the ready state for the first time.
a) Wait time
b) Response time
c) Turnaround time
d) None of these responses are correct.

9. Let S and Q be two semaphores initialized to 1, where P0 and P1 processes the following statements wait(S);wait(Q); ---; signal(S); signal(Q) and wait(Q); wait(S);---;signal(Q);signal(S); respectively. The above situation depicts a _____ .

a) Semaphore
b) Deadlock
c) Signal
d) Interrupt

10. Two short methods that implement the simple semaphore wait() and signal() operations on global variable S include:

```
signal (S) {
  S++;
}
```

and _____ .

a)
```
wait (S) {
 while (S <= 0);
 S--;
}
```

b)
```
wait (S) {
 while (S >= 0);
 S--;
}
```

c)
```
wait (S) {
 S--;
 while (S <= 0);
}
```

d) None of these are correct solutions.

11. A weighted exponential average is useful in process scheduling for:

a) Determining the length of a quantum for the thread.
b) Allowing a priority scheduler to assign a priority to the process.
c) Estimating the length of the next CPU burst for the thread.
d) Ordering processes in the ready queue for a round robin scheduler.

12. Which of the following scheduling algorithms is nonpreemptive?

a) Shortest-Job-First
b) Round-robin
c) First-come-First-Serve
d) Priority algorithms

13. In round-robin scheduling, the time quantum should be _____the context-switch time.

a) small with respect to
b) large with respect to
c) the same size as
d) irrelevant to

14. If the time quantum is very large, a RR (Round-Robin) scheduling is the same as:

a) FCFS
b) SJF
c) SRTN
d) multilevel queue

15. Which of the following is true of multilevel queue scheduling?
a) Processes can move between queues.
b) Each queue has its own scheduling algorithm.
c) A queue cannot have absolute priority over lower-priority queues.
d) It is the most general CPU-scheduling algorithm.

16. A set of processes is _____ when each process in the set is blocked awaiting an event that can only be triggered by another blocked process in the set.
a) spinlocked
b) inactive
c) preempted
d) deadlocked

17. One necessary condition for deadlock is ____, which states that at least one resource must be held in a non-sharable mode.
a) hold and wait
b) mutual exclusion
c) circular wait
d) no preemption

18. A cycle in a resource-allocation graph is ____.
a) a necessary and sufficient condition for deadlock in the case that each resource has more than one instance
b) a necessary and sufficient condition for a deadlock in the case that each resource has exactly one instance
c) a sufficient condition for a deadlock in the case that each resource has more than once instance
d) is neither necessary nor sufficient for indicating deadlock in the case that each resource has exactly one instance

19. One way to ensure that a circular-wait condition never holds is to _____ ?
a) apply a deadlock prevention policy.
b) impose a total ordering of all resource types and to require that each process requests resources in an increasing order of enumeration.
c) assign each resource type a unique integer number to distinguish those occurring at the same time in the ordering.
d) None of these responses is correct.

20. A claim edge of a resource allocation graph _____ .
a) is identical to a request edge
b) resembles an allocation edge in direction but is represented in the graph by a dashed line
c) indicates that a process may request a resource at some time in the future
d) requires that the claiming process be given high priority in scheduling

Question2 (20 points) (*ABET outcome c: Ability to design a system, component, or process to meet desired needs.*)

We have two processes which each repeatedly execute two sections of code, and then increment a shared variable:

```
shared Integer numIterations = 0;

shared Semaphore mutex=1, a1Done=0, b1Done=0;
```

Process A

```
loop begin
      A1;

      signal(a1Done);
      Wait(b1Done);

      A2;

      wait(mutex);
      numIterations++;
      signal(mutex);

loop end;
```

Process B

```
loop begin
      B1;

      signal(b1Done);
      wait(a1Done);

      B2;

      wait(mutex);
      numIterations++;
      signal(mutex);

loop end;
```

We want to satisfy the following constraints:

a) Statement A2 in the *ith* iteration of A's loop cannot execute until statement B1 executes in the ith iteration of B's loop
b) Statement B2 in the ith iteration of B's loop cannot execute until statement A1 executes in the ith iteration of A's loop
c) numIterations must always maintain the number of loops process A has completed plus the number of loops process B has completed

Add to the existing code to satisfy the given constraints, but without adding additional constraints (for example, it shouldn't matter whether A1 or B1 executes first).

You may declare additional shared or local variables of type Integer, Boolean, or Semaphore, but make sure to give them initial values.

Question3 (30 points) (**ABET outcome** *a: Ability to apply mathematics, science and engineering principles.*)

The following table lists the arrival time, execution time, and priority (higher number means greater priority) of 5 jobs.

Job	Arrival time	Execution time	Priority
A	0	30	3
B	20	40	5
C	30	30	4
D	60	20	1
E	100	60	2

Give the start time (the time the job is first scheduled; note that a job may have to wait when it arrives), the end time, the wait time, and the turnaround time of each of the jobs using each of the following scheduling algorithms.

(a) Shortest Job First (without preemption)

Job	Start time	End time	Turnaround	Wait
A	0	30	30	0
C	30	60	30	0
D	60	80	20	0
B	80	120	100	60
E	120	180	80	20
Average job waiting time				80/5=16

(b) Priority (without preemption)

Job	Start time	End time	Turnaround	Wait
A	0	30	30	0
B	30	70	50	10
C	70	100	70	40
E	100	160	60	0
D	160	180	120	100
Average job waiting time				150/5=30

(c) Round-Robin with a quantum of 20

Job	Start time	End time	Turnaround	Wait
A	0	90	90	60
B	20	110	90	50
C	40	120	90	60
D	60	80	20	0
E	120	180	80	20
Average job waiting time				190/5=38

Question 4 (20 points) (*ABET outcome c: Ability to design a system, component, or process to meet desired needs.*)

Consider the following snapshot of a system:

	Allocation				***Max***				***Need***				***Available***			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	1	0	3	2	1	0	5	3	**0**	**0**	**2**	**1**	4	3	1	1
P1	2	1	1	1	7	2	1	4	**5**	**1**	**0**	**3**				
P2	1	1	4	0	1	1	5	0	**0**	**0**	**1**	**0**				
P3	1	0	2	1	3	5	2	2	**2**	**5**	**0**	**1**				
Total	**5**	**2**	**10**	**4**												

a) How many resources are there of type (A, B, C, D)?
***Allocated* + *Available* = 9, 5, 11, 5**

b) Fill in the contents of the matrix *Need* for each process in the space above.

c) Is the system is a safe state? Show why or why not.
Yes the system is safe since there is a safe sequence where all process will be able to finsh with existing resources. One safe sequence is <P2, P0, P1, P3>

P0: 0 0 2 1 > 4 3 1 1 → cannot finish
P1: 5 1 0 0 > 4 3 1 1 → cannot finish
P2: 0 0 1 0 < 4 3 1 1 → can finish and release *Allocation*

4 3 1 1
1 1 4 0

***Available* 5 4 5 1**

P3: 2 5 0 1 > 5 4 5 1 → cannot finish
P0: 0 0 2 1 < 0 0 2 1 → can finish and release *Allocation*

5 4 5 1
1 0 3 2

***Available* 6 4 8 3**

P1: 5 1 0 3 < 6 4 8 3 → can finish and release *Allocation*

6 4 8 3
2 1 1 1

***Available* 8 5 9 4**

P3: 2 5 0 1 < 8 5 9 4 → can finish and release *Allocation*

8 5 9 4
1 0 2 1

***Available* 9 5 11 5**