Q	ABET	Max	Earned
Q1	e	18	
Q2	e	20	
Q3	a	15	
Q4	c	20	
Q5	c	18	
Q6		16	
∑		107	

جامعة بيرزيت

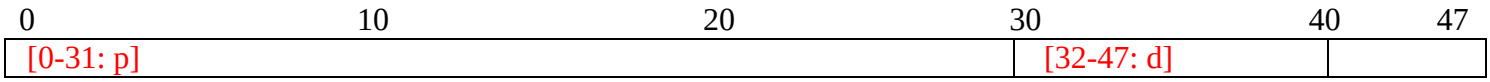
BIRZEIT UNIVERSITY
Electrical and Computer Engineering Department
ENCS339 Operating Systems 2ed Semester 2015/2016
Midterm Exam Instructor: Dr. Adnan H. Yahya Time: 90min

Student Number: **Student Name**

Please answer all questions using the exam sheets ONLY and be BRIEF.
Please show all steps of your solutions. Max grade is:107 (out of 100).
This is a sample solution. Variations will be treated with care and tolerated as long as they are reasonable.

Question 1 (18%) A computer system has only 64GB of **physical memory (RAM)**. The system has a 64KB **page size** and 48-bit logical address space. CPU generated addresses are 6 bytes each. Frame numbers are 4 bytes each.

(a) Indicate on the diagram below which of the bits of the **logical address** of 48 bits are used for page number (**p**) and for offset (**d**)



(b)6% b-1. How many **frames** are there in the RAM?

RAM size in pages = memory size/page size=64GB=2^36B=2^36/2^16=2^20=1M Frames

b-2. If we ignore page table (PMT) overhead and OS needs, how many **pages** can a process have (max) to be runnable in **contiguous** memory allocation mode?
1M Pages (as in b-1)

b-3. How many bits are minimally needed for frame numbers of this computer page map tables (PMTs)?

20 bits to address 1M frames

(c)3% given a 4GB Process what is the size of the Page Map Table (PMT) in **bytes** and **pages** if the PMT is flat (one level)?

4GB=2^32B=2^32/2^16=2^16=64KPages.
64KPages need 64K entries 4 bytes each =256KB.
This is the size of the PMT in Bytes.
=4Pages.

(d)3% Using **two level** paging, find the maximum size (address space, in bytes) that a job **can** have?

Each page can address 16K frames.
2 levels will yield16Kx16k=256M Pages =2^28pages=2^42B=4TB

(e)6% How many levels of page tables would be required to map a full 48 bit virtual address space if the elements in every level of the PMT must fit in a single page? Explain.
48 bits give 256TB
3 levels (that is enough for 2^58)

Question 2 (20%) Consider a computer system involving 5 processes (P1, P2, P3, P4, P5) and 4 different types of resources (R1,R2,R3,R4). The state of the processes and resources is reflected in the tables below.

Currently Available Resources			
R1	**R2**	**R3**	**R4**
2	~~**1**~~**→0**	~~**2**~~**→0**	**0**

	Current Allocation				Max Need				Still Needs			
Process	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	0	0	1	2	0	0	3	2	0	0	2	0
P2	2	0	0	0	2	7	5	0	0	7→**6**	5→**3**	0
P3	0	0	3	4	6	6	5	6	6	6	2	2
P4	2	3	5	4	4	3	5	6	2	0	0	2
P5	0	3	3	2	0	6	5	2	0	3	2	0

(a)8% Use Banker's algorithm to check if this system is currently deadlocked, or can any process become deadlocked if it continues working from the current state? Why or why not? If not deadlocked, give an execution order

Dead-locked □ NO

If Not deadlocked: **Execution Order is:** ___P1,P4,P5,P2,P3_____

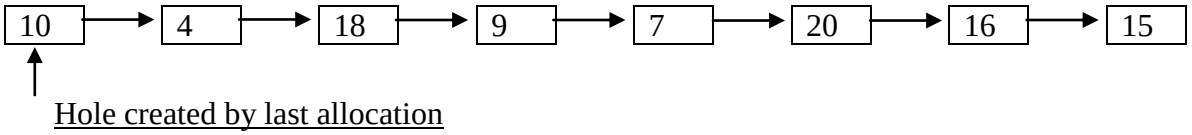
(b)6% If a request from a process P1 asks for the resource vector (0, 4, 2, 0), can the request be immediately granted? Why or why not? If yes, show an execution order. Explain your answer (Briefly).

No. It would have exceeded its max needs in R2. (we don't have these resources)

(c)6% If instead of (b), process P2 asks for the resource vector (0, 1, 2, 0), can the request be immediately granted? Why or why not? If yes, show an execution order. Briefly explain your answer.

No. No process can proceed if P2 is granted these resources. The state will be unsafe. Don't grant the request.

Question 3 (15%) Consider a dynamic partitioning system in which the (free) memory consists of the following list of **holes** (free partitions), sorted by increasing memory address (all sizes are in Megabytes):

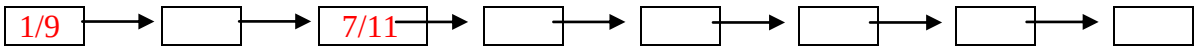


Hole created by last allocation

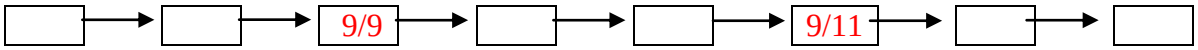
Suppose a new process Pa requiring 11 MB arrives, followed by a process Pb needing 9MB of memory. Show the list of holes **after both of** these processes are placed in memory for each of the following algorithms (start with the original list of holes for each algorithm).

i) First Fit-5%:

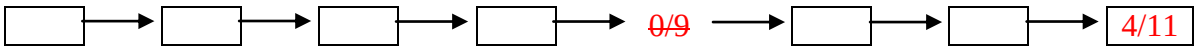
[we show only changed partitions: x/y means x remains after allocating y]: rest unchanged. x+y is original partition size.



ii) Worst Fit -5%:



iii) Best Fit-5%: One partition less:



Question 4 (20%) Consider the following process arrival, CPU burst (in milli-seconds) and explicit **priorities** of the processes A, B, C, D and E. Assume that **5** represents a higher priority than 1.

1. 10% Applying Round Robin combined with priority scheduling algorithm, draw the Gantt charts for both nonpreemptive and preemptive versions and compute the turnaround and wait times and fill the table entries. Assume that quantum = 2ms.

Process	Arrival time	CPU burst time	Priority	Priority/NP			Priority/P			FCFS			SJF			SRTF		
				F	TA	W	F	TA	W	F	TA	W	F	TA	W	F	TA	W
A	0	10	1	48	48	38	48	48	38	10	10	0	10	10	0	10	10	0
B	14	10	1	50	36	26	50	36	26	50	36	26	25	11	1	25	11	1
C	0	10	5	24	24	14	24	24	14	20	20	16	35	35	25	35	35	25
D	2	15	5	30	28	13	30	28	13	35	33	18	50	48	33	50	48	33
E	7	5	5	21	14	9	21	14	9	40	33	28	15	8	3	15	8	3
Avge					30	17.6		30	17.6		26.4	17.6		22.4	10.4		22.4	10.4

(a) nonpreemptive version:

Time	0 2 4 6 8 0 2 4 6 8 0 2 3 5 7 9 0 2 4 6 8 0 2 4 6 8 0
Process	C D C D E C D E C D eC D D D dA B A B A B A B A B

(b) preemptive version:

Time	0 2 4 6 8 0 2 4 6 8 0 2 3 5 7 9 0 2 4 6 8 0 2 4 6 8 0
Process	C D C D E C D E C D eC D D D dA B A B A B A B A B

2. 10% Do the same as above for each of the three scheduling algorithms listed below. Resolve ties alphabetically.

(a) FCFS (First Come First Served).

Time	10	20	35	40	50
Process	A	C	D	E	B

(b) SJF (Shortest Job First).

Time	10	15	25	35	50	
Process	A	E	B	C	D	

(c) SRTF (Shortest Remaining Time First).

Time	10	15	25	35	50	
Process	A	E	B	C	D	

Question 5 (18%)

a. 9% Three threads (A, B, and C) cooperate to sort the contents of an array x of size N as follows. **A** sorts the even numbered elements, x[0], x[2],... . **B** sorts the odd numbered elements, x[1], x[3],.... **C** merge sorts the results of **A** and **B**.
Write a pseudo-code algorithm that handles the coordination between 3 threads. The algorithm must take into account the following:
- Each thread terminates after finishing its work; they do not wait for other threads to complete their execution.
- **A** and **B** are able to access different entries of the array concurrently.
- Use semaphore as a synchronization mechanism.

Regular sort, a function for both A and B.
Only after A and B conclude we start C.
A should issue a wait(s) when it starts (s ant are Semaphores)
B should issue a wait (t) when it starts
Both **s** and **t** are initialized to 1 and released (signal) on conclusion of partial sort.
C issues a wait(s) and wait(t) before it begins the merge sort.

b. 9% Given the following instruction groups A and B). BALANCE initially =300.

```
Scenario 1:
     A1. LOAD R1, BALANCE
     A2. SUB R1, 100
     A3. STORE BALANCE, R1
       Context Switch!
     B1. LOAD R1, BALANCE
     B2. SUB R1, 200
     B3. STORE BALANCE, R1
  a. What is the final value of BALANCE?
BALANCE= _____0_____

Scenario 2:
     A1.  LOAD R1, BALANCE
     A2. SUB R1, 100
       Context Switch!
     B1. LOAD R1, BALANCE
     B2. SUB R1, 200
       Context Switch!
     A3. STORE BALANCE, R1
       Context Switch!
     B3. STORE BALANCE, R1

  b. What is the final value of BALANCE?
 BALANCE= _____100_____
```

c. What is your conclusion regarding the existence of a race in the system (2 lines at most)?

There is a race: the results changes with the order of execution.

d. Use a proper mechanism to synchronize A and B so as to avoid races.

Lock Balance on entry into A (or B) and release on conclusion. The other process waits until Balance lock is released.

Question 6 (16%) True or false

1. □**True ------:** An operating system is a program that acts as an intermediary between the user of a computer and the computer hardware

2. □**True ------:** A user-level process cannot modify its own page table entries

3. ------ □**False:** Context switch time on modern hardware is small enough to be ignored entirely when designing a CPU scheduler.

4. □**True ------:** Setting the program counter register is a privileged operation.

5. ------□**False:** Deadlock is a situation in which two or more processes (or threads) are waiting for an event that will occur in the future.

6. □**True ------:** Races happen in programs when the final results are affected by the execution order of processes.

7. ------ □**False:** Shortest-job-first scheduling is not suitable for a general-purpose computer system.

8. □**True ------:** Generally, each user thread gets assigned to a kernel thread to be run.

9. ------ □**False:** In a multiprocessor system with enough CPUs (cores) a process gets assigned to a given processor (core) to avoid context switches.

10. □**True :** Paging avoids the problem of external fragmentation of memory in a multi-programming environment but has internal fragmentation.

11. ------ □**False:** A process can move form a **ready** state to the **waiting** state, say if a device in its needs set becomes available.

12. □**True ------:** Some kernel-scheduled threads of a process share the same virtual address space

13. □**True ------: I**n symmetric multiprocessor systems threads can not always be run on any processor.

14. □**True ------:** An atomic operation is a machine instruction or a sequence of instructions that must be executed to completion without interruption.

15. □**True ------:** Shortest Job First and Priority scheduling algorithms can lead to starvation?

16. □**True ------:** A resource **lock** is a special case of counting semaphores.