OS

Midterm#1 shalabi

Part 1:

Question 1
Complete
Marked out of 10.00
Flag question

(10 marks/8minutes):4 Processes (P1,P2,P3) arrived at time 0. The Duration of the processes are (2,4,6,8). Each process has a 40% wait time.

Which process finishes first and at which time.

40% I/O Wait Time

If we ignore collisions, access to a hashed file is as fast as an array.

Processes	CPU Utilization
0	0
1	0.6
2	0.70
3	0.80
4	0.90

Answer: A

Select one:

- a. None of the mentioned
- b. P2 at time around 8.88
- c. P3 at time around 8.88
- d. P2 at time around 6.44
- e. P1 at time around 8.88
- f. P1 at time around 6.44
- g. P3 at time around 7.88

Question **2**

Complete

Marked out of 25.00

Flag question

(25 marks/20minutes):Assumptions: All time slice-based algorithms have a time slice of one unit;

The currently running thread is not in the ready queue while it is running;

An arriving thread is run at the beginning of its arrival time, if the scheduling policy allows it.

Turnaround time is defined as the time a process takes to complete after it arrives.

Fill in ALL blanks in EACH table – each blank has an unambiguous answer.

For the missing schedulers, the possibilities are SJF, RR, and Priority. Priority is a preemptive scheduler. Smaller numbers, higher priority (more important).

Process#	Priorities	Arrival Times	Duration (time Unites)
A	6	1	3
B	5	2	3
C	4	5	1
D	3	8	3

Time>	1	2	3	4	5	6	7	8	9	0
FCFS 1	C(1,1)	C(1,2)	C(1,3)	C(1,4)	C(1,5)	C(1,6)	C(1,7)	C(1,8)	C(1,9)	C(1,0)
LJF 2	C(2,1)	C(2,2)	C(2,3)	C(2,4)	C(2,5)	C(2,6)	C(2,7)	C(2,8)	C(2,9)	C(2,0)
Priority 3	C(3,1)	C(3,2)	C(3,3)	C(3,4)	C(3,5)	C(3,6)	C(3,7)	C(3,8)	C(2,9)	C(3,0)

(3,5)	C(3,6)	C(3,7)	C(3,8)	C(2,9)	C(3,0)

Please answer the below questions (list processes for each C(i,j) for each time slot and compute average TurnAround -TA- time).

Example: If the Gantt Chart is

Alg 1	A	A	B	B	D	D	D	A	B	C

Then for

Alg 1	C(1,1)	C(1,2)	C(1,3)	C(1,4)	C(1,5)	C(1,6)	C(1,7)	C(1,8)	C(1,9)	C(1,0)

The answer is: **A,A,B,B,D,D,D,A,B,C** (lists the elements of Gantt chart in order).

FCFS 1	C(1,1)	C(1,2)	C(1,3)	C(1,4)	C(1,5)	C(1,6)	C(1,7)	C(1,8)	C(1,9)	C(1,0)

A,A,A,B,B,B,C,D,D,D

SJF 2	C(2,1)	C(2,2)	C(2,3)	C(2,4)	C(2,5)	C(2,6)	C(2,7)	C(2,8)	C(2,9)	C(2,0)

None of the listed

Average Turnaround Time: Priority (format: x.y: eg: 2.9)(approximate) None of the listed

Average Turnaround Time: FCFS (format: x.y: eg: 2.9)(approximate) None of the listed

Average Turnaround Time: SJF(format: x.y: eg: 2.9) (approximate) None of the listed

Priority 3	C(3,1)	C(3,2)	C(3,3)	C(3,4)	C(3,5)	C(3,6)	C(3,7)	C(3,8)	C(2,9)	C(3,0)

A,A,A,B,B,B,C,D,D,D

Part 2:

Q1: (2 marks/1.5 minutes each) **The scheduler is the part of an Operating System that determines the priority of each process.**

Write one line of explanation.

Select one:

True **False**

Q2: (2 marks/1.5 minutes each) **Rendezvous is a form of messaging that uses indirect addressing.**

Write one line of explanation.

Select one:

True False

Q3:

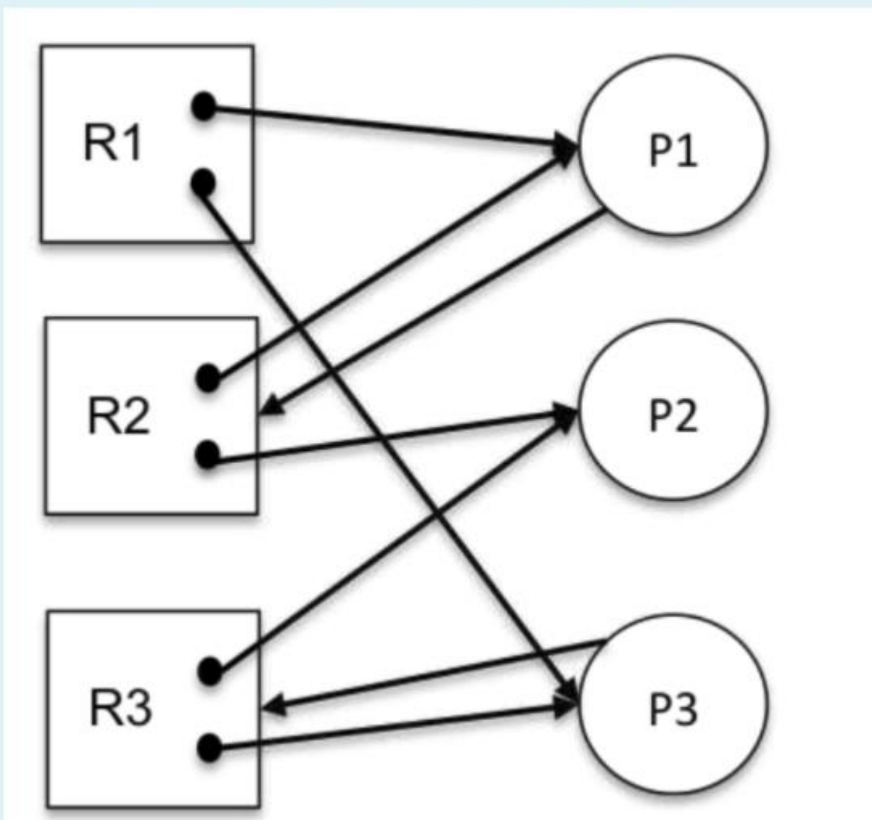
Question 3

Complete

Marked out of 10.00

Flag question

(10 marks/7minutes):Consider the following resource allocation graph (Figure 1)



Select one or more:

a. **Assume now that P2 also demands (requests) recourse R1. The new allocation graph contains a deadlock.**

b. **Figure 1 is NOT deadlock and possible finish order is P2, P1, P3.**

c. Figure 1 is NOT deadlock and possible finish order is P3, P1, P2.

d. Assume now that P2 also demands (requests) recourse R1. The new allocation graph has NO deadlock.

e. Figure 1 is currently deadlock.

Q4: (20 points/15 minutes) **Explain what deadlock detection is and apply it to the following example.**

There are five processes (A, B, C, D and E) and four types of recourses.

Recourses are assigned as follows:

A: (3, 0, 1, 0), B: (0, 1, 0, 0), C: (1, 1, 1, 0), D: (1, 1, 0, 1) and E: (0, 0, 0, 0).

The additional and final request are:

A: (1, 1, 0, 1), B: (0, 1, 1, 2), C: (3, 1, 0, 0), D: (0, 0, 1, 0) and E: (2, 1, 1, 0).

The recourse availability vector is (1, 0, 2, 1).

Determine which requests CANNOT be granted immediately (all that apply).

Select one or more:

1. **C**
2. **A**
3. **E**
4. **B**
5. D

Q5: (2 marks/1.5 minutes each) **A scheduler favoring I/O-bound processes usually does not significantly delay the completion of CPU-bound processes.**

Write one line of explanation.

Select one:

True False

Q6: (2 marks/1.5 minutes each) **Deadlock prevention usually results in poorer (less) utilization of recourse than deadlock avoidance.**

Write one line of explanation.

Select one:

True False

Q7: (2 marks/1.5 minutes each) **Races happen in processes when the final result is independent of the execution order.**

Write one line of explanation.

Select one:

True **False**

Q8: (2 marks/1.5 minutes each) **Threads that are part of the same process share the same stack, heap and code.**

Write one line of explanation.

Select one:

True **False**

Part 3:

Q1: (1.5 marks/1.5 minutes each) **Shared Memory is form of messaging that uses indirect addressing.**

Select one:

True False

Q2: (1.5 marks/1.5 minutes each) **Shortest Remaining Time First and Priority scheduling algorithms can lead to starvation.**

Select one:

True False

Q3: (1.5 marks/1.5 minutes each) **Context switch time is an overhead that is better minimized for better operation of the computer.**

Select one:

True False

Q4: (4 marks/2 minutes each) **Match each scheduler with the task it usually preforms for an operation system,**

Medium Term Scheduler: **Control the degree of Multiprogramming.**

Long Term Scheduler: **Selects a process to be admitted to the system (to ready queue).**

Short Term Scheduler: **Selects a process to be run on the CPU.**

Q5: (10 marks/7 minutes) **Given the following fragment to deal with critical section problem (Pi has a similar construct):**

Process Pj:

```
do {
      flag[j] = true;
```

```
        turn = j;
        while(flag[i] && turn == i);
                // critical section
                flag[j] = false;
        // remainder section

} while(true);
```

Select one:

1. **This is wrong solution as both Pi and Pj can enter their critical sections.**
2. This is correct solution as Pi and Pj can enter its critical section in turn (one then the other and so on).
3. This is correct solution as only one of Pi and Pj can enter its critical section at a time.
4. This is wrong solution as neither Pi nor Pj can enter their critical sections.

Q6: (1.5 marks/1.5 minutes each) **Threads are cheaper to context switch than processes.**

Select one:

True False

Q7: (4 marks/2 minutes) **In contrast to cooperative scheduler, a preemptive scheduler supports the following state transition:**

Select one:

1. Ready ⇨ Running
2. **Running ⇨ Ready**
3. Blocked ⇨ Running
4. Ready ⇨ Blocked

Q8: (1.5 marks/1.5 minutes each) **A process can hold only one lock at a time.**

Select one:

True False

Q9: (1.5 marks/1.5 minutes each) **A multilevel feedback queue scheduler generally assigns a long quantum to:**

Select one:

1. High priority processes
2. **Low priority processes**
3. Older processes
4. New processes

Q10: (1.5 marks/1.5 minutes each) **A SJF scheduler may preempt an already running longer job.**

Select one:

True **False**

Midterm#2

Part 1:

Q1:

(25 marks/20minutes):Assumptions: All time slice-based algorithms have a time slice of one unit;

The currently running thread is not in the ready queue while it is running;

An arriving thread is run at the beginning of its arrival time, if the scheduling policy allows it.

Turnaround time is defined as the time a process takes to complete after it arrives.

Fill in ALL blanks in EACH table – each blank has an unambiguous answer.

For the missing schedulers, the possibilities are SRTF, RR, and Priority. Priority is a preemptive scheduler. Larger numbers, higher priority (more important).

Process#	Priorities	Arrival Times	Duration (time Unites)
A	3	1	3
B	4	2	3
C	5	5	1
D	6	8	3

Time>	1	2	3	4	5	6	7	8	9	0
FCFS 1	C(1,1)	C(1,2)	C(1,3)	C(1,4)	C(1,5)	C(1,6)	C(1,7)	C(1,8)	C(1,9)	C(1,0)
SRTF 2	C(2,1)	C(2,2)	C(2,3)	C(2,4)	C(2,5)	C(2,6)	C(2,7)	C(2,8)	C(2,9)	C(2,0)
Priority 3	C(3,1)	C(3,2)	C(3,3)	C(3,4)	C(3,5)	C(3,6)	C(3,7)	C(3,8)	C(2,9)	C(3,0)

(3,5)	C(3,6)	C(3,7)	C(3,8)	C(2,9)	C(3,0)

Please answer the below questions (list processes for each C(i,j) for each time slot and compute average TurnAround -TA- time).

Example: If the Gantt Chart is

Alg 1	A	A	B	B	D	D	D	A	B	C

Then for

Alg 1	C(1,1)	C(1,2)	C(1,3)	C(1,4)	C(1,5)	C(1,6)	C(1,7)	C(1,8)	C(1,9)	C(1,0)

The answer is: **A,A,B,B,D,D,D,A,B,C** (lists the elements of Gantt chart in order).

Priority 3	C(3,1)	C(3,2)	C(3,3)	C(3,4)	C(3,5)	C(3,6)	C(3,7)	C(3,8)	C(2,9)	C(3,0)

A,B,B,B,C,A,A,D,D,D

Average Turnaround Time: FCFS (format: x.y: eg: 2.9) (approximate) 3.5

Priority 3	C(3,1)	C(3,2)	C(3,3)	C(3,4)	C(3,5)	C(3,6)	C(3,7)	C(3,8)	C(2,9)	C(3,0)

A,B,B,B,C,A,A,D,D,D

Average Turnaround Time: FCFS (format: x.y: eg: 2.9) (approximate) 3.5

Average Turnaround Time: SRTF(format: x.y: eg: 2.9) (approximate) 3.25

FCFS 1	C(1,1)	C(1,2)	C(1,3)	C(1,4)	C(1,5)	C(1,6)	C(1,7)	C(1,8)	C(1,9)	C(1,0)

A,A,A,B,B,B,C,D,D,D

SRTF 2	C(2,1)	C(2,2)	C(2,3)	C(2,4)	C(2,5)	C(2,6)	C(2,7)	C(2,8)	C(2,9)	C(2,0)

A,A,A,B,C,B,B,D,D,D

Average Turnaround Time: Priority (format: x.y: eg: 2.9) (approximate) 3.5

Q2:

(10 marks/8minutes): 4 Processes (P1,P2,P3,P4) arrived at time 0. The Duration of the processes are (6,8,4,2). Each process has a 50% wait time.
Which process finishes first and at which time.

50% I/O Wait Time

Processes	CPU Utilization
0	0
1	0.5
2	0.75
3	0.875
4	0.9375

Answer: F

Select one:

- a. None of the mentioned
- b. P2 at time around 3.43
- c. P2 at time around4.27
- d. P4 at time around 3.43
- e. P1 at time around 3.43
- f. P4 at time around4.27
- g. P1 at time around 4.27

Part 2:

Q1: (2 marks/1.5 minutes each) **A job always makes at least some progress when scheduled by the short-term dispatcher.**

Write one line of explanation.

Select one:

True **False**

Q2: (20 points/15 minutes) **Explain what deadlock detection is and apply it to the following example.**

There are five processes (A, B, C, D and E) and four types of recourses.

Recourses are assigned as follows:

A: (3, 0, 1, 0), B: (0, 1, 0, 0), C: (1, 1, 1, 0), D: (1, 1, 0, 1) and E: (0, 0, 0, 0).

The additional and final request are:

A: (1, 1, 0, 1), B: (6, 1, 1, 1), C: (3, 1, 0, 0), D: (0, 0, 1, 0) and E: (0, 1, 1, 2).

The recourse availability vector is (1, 0, 1, 0).

Determine whether the current state is SAFE and select the right answer below.

Select one or more:

1. Safe and the order is D⇨E⇨C⇨B⇨A.
2. Not safe and no process can progress.
3. **Not safe but D⇨A⇨C⇨B can work.**
4. Safe and the order is D⇨A⇨C⇨B⇨E.

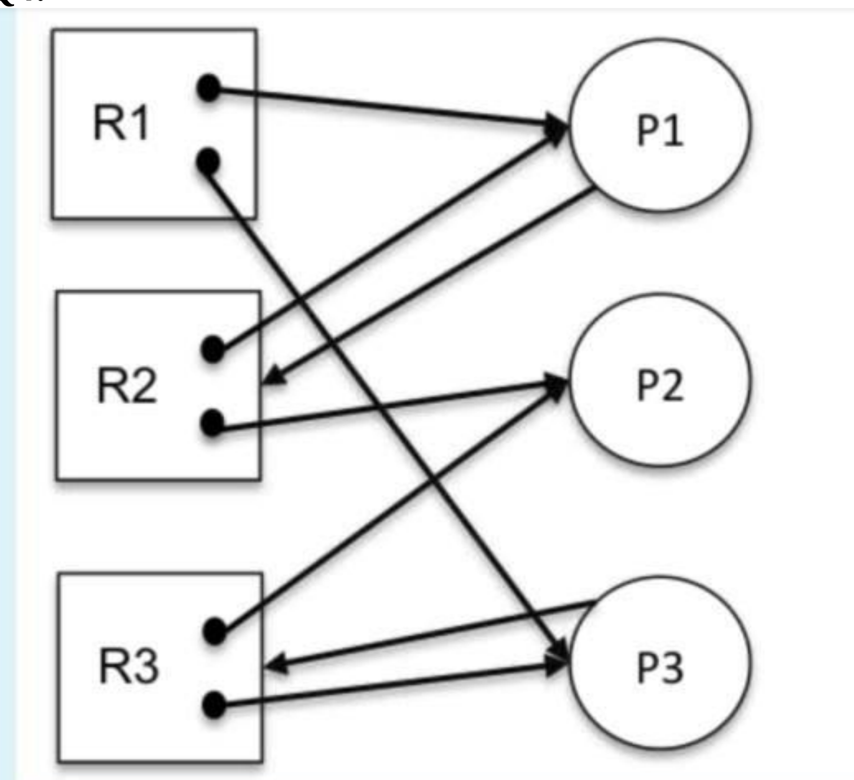
Q3: (2 marks/1.5 minutes each) **Longest Remaining Time First and Priority scheduling algorithms can lead to starvation.**

Write one line of explanation.

Select one:

True False

Q4:



Select one or more:

- [ ] a. Figure 1 is currently deadlocked.
- [ ] b. Figure 1 is NOT deadlocked and a possible finish order is P3,P1,P2
- [x] c. Figure 1 is NOT deadlocked and a possible finish order is P2,P1,P3
- [ ] d. Assume now that P2 also demands (requests) resource R1. The new allocation graph has No deadlock.
- [x] e. Assume now that P2 also demands (requests) resource R1. The new allocation graph contains a deadlock.

Part 3:

Q1: (4 marks/2 minutes) **When does preemption take place?**

Select one:

1. **When a quantum expires.**
2. When a process issues an I/O request.
3. When a process exits.
4. All of the mentioned.

Q2: (2 marks/1.5 minutes each) **Deadlock detected and recover usually results in poorer (less) utilization of resource than deadlock avoidance.**

Select one:

True False

Q3: (2 marks/1.5 minutes each) **In a multicore system all local core memories are equally accessible to all cores, which is not the case for Multi-Processor systems.**

Select one:

True False

Q4: (2 marks/1.5 minutes each) **Given a stream of jobs (processes) that arrive at the same time and have different priorities each, Preemptive and NonPreemptive priority scheduling always result in the same average Wait Time for the stream.**

Select one:

True False

Q5: (2 marks/1.5 minutes each) **The Gantt Chart P1 P2 P3 P1 P2 P3 P1 result from RR scheduling but not from SJF scheduling.**

Select one:

True False

Q6: (2 marks/1.5 minutes each) **An Operating System is a program that acts as an intermediary between the computer hardware and the user of a computer. That’s why users cannot access computer RAM directly.**

Select one:

True False

BIRZEIT UNIVERSITY
Electrical Engineering Department
ENCS339 Operating Systems

Second Semester, 2018-2019 — Instructors: Dr. Adnan H. Yahya
Final Exam — Time 150 minutes (2.5 Hours)

Question ABET SO	Q1 a	Q2	Q3 c	Q4 e	Q5	Q6	**Total**	**Student Name**
Grade								
Max	12	15	15	15	15	36	**108**	**Student number**

Please answer all questions using the provided exam sheets only. Max Grade 105.

Question 1 (12%): List the order in which the following 5 requests: 32, 98, 112, 52, 190 for a given cylinder number will be serviced for each of the different disk scheduling algorithms. There are 200 cylinders numbered from 0 – 199. The disk head starts at number **100** and when needed assume the head is moving outward (towards 200). Find the time per the sequence assuming a cost of 1 for each cylinder travel and the average per the entire sequence. Show all solution steps.

1- ***FCFS*** – *first come first served*: In order of arrival.

```
                                   v→
|     |     |     |     |     |     |     |     |     |     |
0     20    40    60    80    100   120   140   160   180   200
```

100→32→98→112→52→190→ |100-32|+|98-32|+|112-98|+|112-52|+|190-52|=
68 + 66 + 14 + 60 + 138=346

Average time per request=346/5=69.2

2- ***SSTF*** – *shortest seek time first'*: Closest to head first

```
                                  v→
|     |     |     |     |     |     |     |     |     |     |
0     20    40    60    80    100   120   140   160   180   200
```

100→98→112→52→32→190→ |100-98|+|112-98|+|112-52|+|52-32|+|190-32|=
2 + 14 + 60 + 20 + 158=254

Average time per request=254/5=50.8

3- ***SCAN*-Look** which means***: Elevator but go back when no more requests in that direction.***

```
                                   v→
|     |     |     |     |     |     |     |     |     |     |
0     20    40    60    80    100   120   140   160   180   200
```

100→112→190→98→52→32→ |112-100|+|190-112|+|190-98|+|98-52|+|52-32|=
12 + 78 + 92 + 46 + 20=250

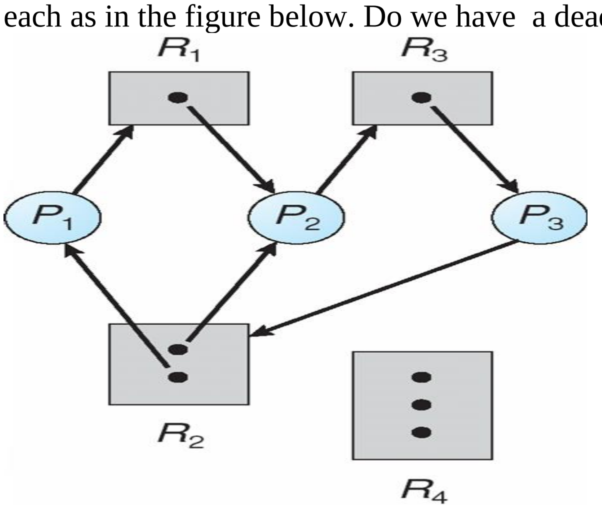
Average time per request=246/5=49.6

Question 2 (15%):a- 11% Match the term in the left column to the definition in the right that fits best:
Fill the table below also with your selections (-2%)

B	(1) Latency Time	~~(A)~~ Process part having a logical interpretation
M	(2) File control block	~~(B)~~ Time for the desired data to spin under the desk head.
J	(3) Compaction	**(C)** processes sharing code but not necessarily data
F	(4) DES	~~(D)~~ A region of code accessing shared information that needs to be protected by mutual exclusion
O	(5) process synchronization	~~(E)~~ Moving Blocks of data between RAM and Disk without Intensive CPU Intervention.
D	(6) Critical section	~~(F)~~ Symmetric Encryption
O	(7) busy waiting	~~(G)~~ Happens when the RAM is much less than the total working set of running processes.
G/E	(8) Thrashing	**(H)** The data structure storing the state of a process including registers, stack, memory protections, time used, etc.
I	(9) Defragmentation	~~(I)~~ Combine individual file parts into a contiguous space.
A	(10) Segment	~~(J)~~ Combine the memory holes into a single memory hole.
L	(11) RSA	**(K)** Time for the head to move to the desired track of the disk.
N	(12) Access Control List	~~(L)~~ Asymmetric Encryption
		~~(M)~~ The data about the file like its change dates, name, ID and storage location on disk,….
		~~(N)~~ Specification of rights to a resource
		(O)- None of the above

Item ➔	1	2	3	4	5	6	7	8	9	10	11	12	#Correct
Matching Letter	**B**	**M**	**J**	**F**	**O**	**D**	**O**	**G/E**	**I**	**A**	**L**	**N**	

(b) 4%. A system has three processes (P1,P2, P3) and 4 resources (R1,R2, R3,R4) with 1, 1,2 and 3 instances each as in the figure below. Do we have a deadlock? Why?



Answer: X Deadlock ☐ No Deadlock

Reason: **No process can progress, circular wait (multiple loops).**

Question 3 (15%) Consider the following page reference string (15 memory references) in a demand paging virtual memory environment (repeated in tables):

1	2	3	4	2	1	5	6	2	1	2	3	7	5	3	2	1	2	3	6

16% Calculate how many page faults would occur, the success rate and failure rate for each of the following replacement algorithms, We have 3 frames F1-F3 and all frames are initially empty.

a. **Optimal (OPT) replacement** (5%)**:**

Page#➔	**1**	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**5**	**3**	**2**	**1**	**2**	**3**	**6**
Frame1	**1**	**1**	**1**	**1**			**1**	**1**				**1**	**7**	**5**			**1**			**6**
Frame2		**2**	**2**	**2**			**2**	**2**				**2**	**2**	**2**			**2**			**2**
Frame3			**3**	**4**			**5**	**6**				**3**	**3**	**3**			**3**			**3**
Fault?	+	+	+	+			+	+				+	+	+			+			+

Success Rate S= 9/20=45 % **Failure Rate F= 11/20=55 %**

b. **LRU replacement** (5%)

Page#➔	**1**	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**5**	**3**	**2**	**1**	**2**	**3**	**6**
Frame1	**1**	**1**	**1**	**4**		**4**	**4**	**6**		**6**		**3**	**3**	**3**		**3**	**3**			**3**
Frame2		**2**	**2**	**2**		**2**	**2**	**2**		**2**		**2**	**2**	**5**		**5**	**1**			**6**
Frame3			**3**	**3**		**3**	**5**	**5**		**1**		**1**	**7**	**7**		**2**	**2**			**2**
Fault?	+	+	+	+		+	+	+		+		+	+	+		+	+			+

Success Rate S=5/20=25 % **Failure Rate F= 15/20=75 %**

3. **FIFO with 3 frames:** (5%)

Page#➔	**1**	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**5**	**3**	**2**	**1**	**2**	**3**	**6**
Frame1	**1**	**1**	**3**	**3**	**2**	**2**	**5**	**5**	**2**	**2**		**3**	**3**	**5**	**5**	**2**	**2**		**3**	**3**
Frame2		**2**	**2**	**4**	**4**	**1**	**1**	**6**	**6**	**1**		**1**	**7**	**7**	**3**	**3**	**1**		**1**	**6**
Fault?	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+		+	+

Success Rate S= 2/20=10 % **Failure Rate F= 18/20=90 %**

Question 4 (15%)

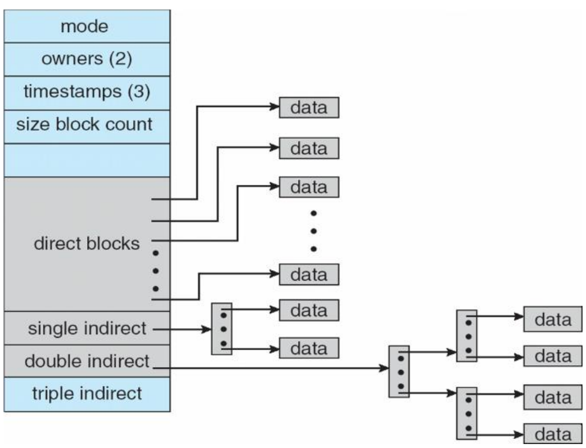
(a) 5% Suppose a computer has a file system for a 128GB disk, where each disk block is 4KB. If the OS for this computer uses a File Allocation Table (FAT), what is the smallest amount of memory that could possibly be used for the FAT (assuming the entire FAT is in memory -RAM-)? Explain.

4KB=2**12; 128GB=2**37; number of blocks= 2**37/2**12=2**25 Blocks:
need 25bits or 4 bytes pointers
2**25x4Bytes=2**27=128MB

(b) 3% In the above file system suppose half of all files are exactly 2KB and the other half of all files are exactly 3KB exactly. What fraction of disk space would be wasted? (Consider only blocks used to store data)? Explain why!

Allocation is always in full blocks of 4KB each.
If size = 2KB and is allocated 4KB: waste = 50%.
If size = 3KB and is allocated 4KB: waste = 25%.
Average wase is (25+50)/2=37.5%.

(C). 7% Suppose that on a different computer, the OS uses UNIX i-nodes as in the figure below and each disk block is 8KB. Assume that an i-node contains 12 direct block numbers (disk addresses) and the block numbers for one indirect block, one double indirect block, and one triple indirect block. Assume also that a block number is 4 bytes. Compute the maximum address space (in bytes) a file can have in this system.



Direct Blocks: 12*8K=96KB.
Single inDirect Block: 8KB can have as many as 2K pointer (8KB/4).
Total size: 2Kblocks*8KB=16MB.

Double inDirect Blocks: As before 8KB can have as many as 2K pointer (8KB/4).
Total size: 2Kblocks*2Kblocks*8KB=32GB.
Triple inDirect Blocks: As before 8KB can have as many as 2K pointer (8KB/4).
Total size: 2Kblocks*2Kblocks*2Kblocks*8KB=64TB.
Total size:= **64TB+32GB+16MB+96KB**.

Question 5 (15%) Disk RAID: Consider that many RAID devices now ship with the following options:

- RAID0 --- data striped Across all disks
- RAID1 --- each disk mirrored
- RAID 5 --- striped parity

Assume a system with 8 disks. **(of capacity X each):**

1- 3% For each level, how much usable storage does the system receive?

RAID0–8 disks
RAID1–4disks
RAID5–7disks

2- 3% Assume a workload consisting only of small reads, evenly distributed. What is the throughput of each level assuming one disk does 100 reads/sec?

RAID0–800 reqs/sec
RAID1–800 reqs/sec
–reads can be satisfied from both disks in a pair
RAID5–800 reqs/sec
–no need to read the parity, so no loss of read performance, only space

3- 3% Assume a workload consisting only of small writes, evenly distributed. Again, calculate the throughput assuming one disk does 100 writes/sec

RAID0–800 reqs/sec
RAID1– 400 reqs/sec
– Need to write to both disks in a pair
RAID5– 200 reqs/sec
If you do two reads + two writes to update the parity, or 100 reqs/sec if you read all of the disks
To recalculate the parity

4- 2% For each level, what is the minimum number of disks that may fail before data **may** be lost?

RAID0–1, but data loss is guaranteed at the first lost disk
RAID1–2, if you happen to lose both disks in a pair
RAID5–2, but data loss is guaranteed on the second disk

5- 2% For each level, what is the minimum number of disks that must fail to **guarantee** data loss?

RAID0---1
RAID1– 5, if you happen to get really lucky and lose one from each pair before losing the 5th
RAID5--- 2

Question 6 (36%): Mark (X) **True** or **False**. Also fill the answer sheets below (Fill the tables: -4% if not)
Add a line of explanation (-5% if not).

1. **XTrue** or □**False** A safe state guarantees that there is an ordering in which all the processes in the system can terminate their operations.
2. □**True** or **XFalse** Regular users should have enough security privileges to meet their needs (least privilege principle) but at least one superuser must have all security privileges to a system to help other users access their data in management approved cases.
3. □**True** or **XFalse** Both starvation and deadlock have the negative side that the CPU is not able to work although some processes are waiting to get access to that CPU.
4. □**True** or **XFalse** Deadlock cannot occur if the number of each resource instances is greater than the MAX need for that resource of the most resource hungry process in the system.
5. **XTrue** or □**False** Given a set of processes that arrived at time 0: using Round Robin for multiprogramming and ignoring context switch time the sum of all TurnAround times of all jobs is the same as the sum of burst times for the processes.
6. □**True** or **XFalse** A user-level process modifies its own page table (PMT) entries, say when it needs more space. The time to do that is overhead time.
7. **XTrue** or □**False** The working set of a job cannot exceed (cannot be more than) the number of pages referenced by the job during its lifetime.
8. **XTrue** or □**False** Last Come First Served Process scheduling is preemptive scheduling.
9. **XTrue** or □**False** Binary semaphore is a special case of counting semaphores.
10. **XTrue** or □**False** Atomic operations may contain multiple instructions but cannot be interrupted before the complete execution of all instructions of the operation.
11. **XTrue** or □**False** The normal page table specifies the frame number for each job page while the Inverted page table gives the process number and page number of that process for each frame.
12. **XTrue** or □**False** It is generally less time consuming to map pages to frames using Inverted tables than using regular PMTs.
13. **XTrue** or □**False** In a flat directory it is not possible to have one file with 2 names.
14. **XTrue** or □**False** round robin (RR) multiprogramming can increase the wait time of a group of processes, as opposed to SJF scheduling.
15. **XTrue** or □**False** There can be cases in Demand paging when increasing memory size can result in increased number of page faults.
16. **XTrue** or □**False** DMA is a mechanism for allowing an I/O device to transfer data to and from memory without involving the CPU in the transfer.
17. □**True** or **XFalse** Memory mapped I/O determines how the pages of an I/O-bound process are mapped to page frames.
18. **XTrue** or □**False** A race happens when different orders of execution result in different final results and it must be avoided through synchronization.
19. □**True** or **XFalse** A context switch from one process to another can be accomplished without executing OS code in kernel mode.
20. **XTrue** or □**False** An advantage of implementing threads in user space is that they don't incur -يكلف- the overhead of having the OS schedule their execution.

Q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
□T	XT	□T	□T	□T	XT	□T	XT	XT	XT	XT	XT	XT	XT	XT	XT	XT	□T	XT	□T	XT
□F	□F	XF	XF	XF	□F	XF	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	XF	□F	XF	□F

Mark (X) where applicable:

21. **XTrue** or □**False** Deadlock can never occur if no process is allowed to hold a resource while requesting another resource.
22. **XTrue** or □**False** In round robin scheduling, it is advantageous to give each I/O bound process a longer quantum than each CPU-bound process (since this has the effect of giving theI/O bound process a higher priority).
23. □**True** or **XFalse** A message coded (encrypted) with a public key of A (EpA) can be decoded (decrypted) with the same public key of user A (EpA).
24. **XTrue** or □**False** A TLB miss could occur even though the requested page was in memory.
25. **XTrue** or □**False** Associative memory in the form of TLBs is used to speed-up page lookup in a paged virtual memory system because it is fast and searching all its entries for the needed page can be done fast for small sized TLBs.
26. **XTrue** or □**False** SPOOLing is an approach to convert sequential access devices to Random/Direct Access devices. For example it is used to have a printer used by multiple processes at the same time.
27. **XTrue** or □**False** Device drivers take care of the distinctive characteristics of input devices and passes standard data from the device to the CPU.
28. **XTrue** or □**False** It is better to store the disk directory on the same device/partition as the data rather than having a central location for the directory for all volumes.
29. **XTrue** or □**False** Only internal fragmentation occurs in paging while both External and Internal fragmentation occur in a purely partitioned memory management.
30. **XTrue** or □**False** If we ignore collisions, access to a hashed file is as fast as an array provided a good hashing function is used.
31. **XTrue** or □**False** In **grouping** and **counting** arrangements for free space disk management some blocks consist of only index pointers while others are completely free and have neither data nor pointers.
32. □**True** or **XFalse** Using general graphs is preferable to (better than) using acyclic graphs as the main data structure of the file directory in terms of supporting all operations.
33. □**True** or **XFalse** In a system with one instance of each resource except one a cycle in the need graph indicates a sure deadlock in the system.
34. Which of the following factors could be used to argue **for** larger page size (Mark one):
    □a. Absence of Internal fragmentation. □b. Smaller Process page table size.
    □c. Better External fragmentation. Xd. Less Thrashing
35. Which is **NOT** an advantage of any asymmetric message encryption(Mark one):
    □**a.** Authentication □**b.** Integrity
    □**c.** Targeted delivery **Xd.** Compression
36. Which is **NOT** a security threat in computing systems (Mark one):
    □**a.** Trap Door □**b.** Trojan Horse
    Xc. Crypto Currency □**d.** Worms

Q	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	#Correct
□T □F	XT □F	XT □F	□T XF	XT □F	XT □F	XT □F	XT □F	XT □F	XT □F	XT □F	XT □F	□T XF	□T XF	□a □b □c Xd	□a □b □c Xd	□a □b Xc □d	

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

BIRZEIT UNIVERSITY
Electrical and Computer Engineering Department
ENCS339 Operating Systems 2nd Semester 2018/2019
Midterm Exam Instructor: Dr. Adnan H. Yahya Time: 90min

Student Name:_____<u>Sample Solution</u>___ Student Number:0000
Please answer all questions using the exam sheets ONLY.
Please show all steps of your solutions. Max grade is:107.

Question 1 (15%) A computer system has 24GB of **physical memory (RAM)**. From historical data we know that the process distribution is as follows:

Size in GB	Percentage of load		**Partition Size (GB)**	**Number of Partitions**
0- 0.25	10%		0.25	X→ 4
0.25- 0.30	25%		0.30	2.5X→10
0.3- 0.50	40%		0.50	4X→16
0.5- 1.00	20%		1.00	2X→8
1.0- 2.00	5%		2.00	0.5X→2

We want to divide the memory into a fixed number of partitions of fixed size. What are the size and number of the partitions in the system. Fill the table above and explain.
9%X*0.25+2.5X*0.30+4X*0.5+2X*1+0.5X*2=24(GB)
X(0.25+0.75+2+2+1)= 24→ X=24/6=4 Partitions
Total number of partitions = 40.
Another solution: If we have 1 for last case: we need to have 4, 8, 5, 2 for the previous cases for a total of
1*2GB+4*1GB+8*0.5GB+5*0.3GB+2*0.25=2+4+4+1.5+0.5GB=12GB
But we have 24GB memory so we can have double: as in the table.

3%Does this arrangement have Internal fragmentation? □**Yes** □**No: Explain or say how much on average.**
On average: the first will have 0.125GB waste, the 2nd will have 0.025 =((0.3-0.25)/2) waste, the 3rd will have 0.1= ((0.5-0.3)/2) waste, the 4th will have 0.25= ((1-0.5)/2) waste, the 5th will have 0.5= ((2-1)/2) waste.
The average waste: 4*0.125+10*0.025+ 16*0.1+8*0.25+2*0.5/40=5.35/40=0.13375GB

3%Does this arrangement have External fragmentation? □**Yes** □**No: Explain or say how much on average.**

When there is a job that is larger than the available partitions:
can say half in the worst case (just a yes answer will do).

Question 2 (18%) Consider a computer system involving 5 processes (P1, P2, P3, P4, P5) and 4 different types of resources (R1,R2,R3,R4). The current state of the processes and resources is reflected in the tables below.

Currently Available Resources			
R1	**R2**	**R3**	**R4**
2	**1**	**2**	**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	5	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) 5% 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 **Deadlocked** □**YES** □ **NO** **2132 4456 4788 6788 67 11 12**
If Not deadlocked: Execution Order is (just add indices): P1➔ P4➔ P5➔ P2➔ P3

(b) 4% If a request from 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.
Request Can be Granted: □**YES** □ **NO**
More than available More than need (either will do)

If granted, Execution Order is (just add indices): P___➔ P___➔ P____➔ P___➔ P___

(c) 4% 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. Explain your answer.
Assume granted: remaining available: 2000, P2 still needs: 0630: P1 can't work, Neither P2, Neither P3, neither P4 neither P5. Unsafe State.
Request Can be Granted: □**YES** □ **NO**

If granted, Execution Order is (just add indices): P___➔ P___➔ P____➔ P___➔ P___

Question 3 (20%: 4% each) Consider a dynamic (contiguous) partitioning system in which the (free) memory consists of the following list of **<u>holes</u>** (free partitions), sorted by increasing memory address (all sizes are in Megabytes) and an arrow pointing to the first partition (<u>Hole List Start Pointer (HLPTR)</u> :

7 → 21 → 8 → 12 → 5 → 13 → 7 → 23

↑ <u>Hole List Start Pointer (HLPTR)</u>

Suppose a new process Pa requiring 12 MB arrives, followed by a process Pb needing 8MB of memory. Show the list of holes **after each of** these processes are placed in memory for each of the following algorithms (start with the original list of holes for each algorithm). Assume that **the hole List Start Pointer always points to the leftmost in the hole.**

i) First Fit:

7 → 9 → 8 → 12 → 5 → 13 → 7 → 23

7 → 1 → 8 → 12 → 5 → 13 → 7 → 23

ii) **Worst Fit** -5%:

7 → 21 → 8 → 12 → 5 → 13 → 7 → 11

→ 13 → 8 → 12 → 5 → 13 → 7 → 11

iii) **Best Fit**-5%:

7 → 21 → 8 → 1 → 5 → 13 → 7 → 23

7 → 21 → 1 → 5 → 13 → 7 → 23 →

iv) **Best Fit Plus 50%**- meaning best fit but each process gets exactly (size +50%) hole:

7 → 3 → 8 → 12 → 5 → 13 → 7 → 23

7 → 3 → 8 → 5 → 13 → 7 → 23 →

Last pointer is nil.

Question 4 (20%, 5% each)

Consider the following set of jobs to be scheduled for execution on a single CPU system.

Job	Arrival Time	Size (msec)	Priority
J_1	0	10	2 (Silver)
J_2	2	8	1 (Gold)
J_3	3	3	3 (Bronze)
J_4	10	4	2 (Silver)
J_5	12	1	3 (Bronze)
J_6	15	4	1 (Gold)

a. Draw a Gantt chart showing **FCFS** scheduling for these jobs.

0 1 10 20 30

Time	0 1 10 20 30
Job	J1J1J1J1J1J1J1J1J1J1J2J2 J2 J2 J2 J2 J2 J2J3 J3 J3J4 J4 J4 J4 J5J6 J6 J6 J6

b. Draw a Gantt chart showing **SRTF** scheduling for these jobs.

Time	0 1 10 20 30
Job	J1J1J1J3J3J3J1J1J1J1J1J1 J1 J5 J4 J4 J4 J4J6 J6 J6J6 J2 J2 J2J2J2J2J2J2

c. Draw a Gantt chart showing **Preemptive Priority** scheduling for these jobs. List priorities from highest to lowest: Highest is 1 GOLD Lowest is: 3 BRONZE

Time	0 1 10 20 30
Job	J1J1J2J2J2J2J2J2J2J2J1J1 J1 J1 J1 J1 J6 J6J6 J6 J1J1 J4 J4 J4J4J3J3J3J5

d. Draw a Gantt chart showing **Shortest Job First (SJF)** scheduling for these jobs.

Time	0 1 10 20 30
Job	J1J1J1J1J1J1J1J1J1J1J3J3 J3 J5 J4 J4 J4 J4J6 J6 J6J6 J2 J2 J2J2J2J2J2J2

e. Which of the foregoing scheduling policies provides the lowest waiting time for this set of jobs? What is the waiting time with this policy? (Show your work)

SJF is the one that gives lowest waiting time: Waiting time= (10-10)+ (28-8)+(9-3)+(8-4)+(2-1)+(7-4)=0+20+6+4+1+3=34 (on average: 34/6=5.67).

Question 5 (18%) Recall the definitions of semaphore operations Wait and Signal:

Wait(S) { while (S <= 0); // busy wait
S--;}
Signal(S) { S++;}

4 semaphores with initial values: semaphore S1=0, S2=0, L1=1, L2=1;		
//Thread 1 1. Wait(L1); 2. Signal(S1); 3. Wait(S2); 4. Wait(L2);		// Thread 2 1. Wait(L2); 2. Signal(S2); 3. Wait(S1); 4. Wait(L1);

Command 2 of thread 1 is 1.2, Command 3 of thread 2 is 2.3 and so on (thread.instruction)

a. Can thread 1 finish ? □**YES** □ **NO** List the commands it executes.

0011→0001→ 1001 → 1000 → 1100

1.1 Wait(L1); 1.2 Signal(S1); 2.1 Wait(L2); 2.2 Signal(S2); ??

Any other sequence will result in the same problem. Deadlocked situation

b. Can thread 2 finish ? □**YES** □ **NO** List the commands it executes.

See above

c. Can the system get deadlocked? □**YES** □ **NO** **Why?**

There is no signaling for Ls. All are waiting and initial value is 1. If initial value is 2 then it may work.

d. Is there an execution order that allows the two threads to finish? □**YES** □ **NO Which?**

Double over integers could be implemented as in that order (SL is Shift Left):

1. register1 = counter
2. register1 = SL register1
3. counter = register1

Half over integers could be implemented as in that order (SR is Shift Right)

1'. register2 = counter
2'. register2 = SR register2
3'. counter = register2

If Counter is initially 8 then we execute Double and Half:

e. Give a sequence when the result is correct: 1→2→3→1'→2'→3'
Or 1'→2'→3'→1→2→3 [result equal original:8]-one finishes before other starts.

f. Give a sequence when the result is incorrect and the resulting value.

g. : 1→2→1'→2'→3'→3 [result = 16] Or 1'→2'→1→2→3→3' [result = 4]
Others available!!!!

Question 6 (20%) True or false, add line of explanation as to WHY. Fill the table (-3% if not)

1. □**True** □**False:** An operating system is a program that acts as an intermediary between the user of a computer and the computer hardware. One definition of OS.
2. □**True** □**False:** A user-level process cannot modify its own page table entries. OS Can
3. □**True** □**False:** In a multiprocessor system with enough CPUs (cores) a process gets assigned to a given processor (core) permanently to avoid context switches. Context switches happens even if assigned to a single core: when more than one process run on that core.
4. □**True** □**False:** A process can move form a **ready** state to the **waiting** state, say if a device it needs becomes available. Only from running on CPU to waiting: not from ready.
5. □**True** □**False: I**n a symmetric multiprocessor, threads cannot always be run on any processor. It can as symmetric means all are equal.
6. □**True** □**False:** Shortest Job First and Priority scheduling algorithms can lead to starvation? Shorter jobs can keep coming and the longer ones have no chance.
7. □**True** □**False:** A SJF scheduler may preempt a previously running longer job. SJF is nonpreemptive: SRTF is preemptive, however.
8. □**True** □**False:** If all jobs have identical run lengths, a RR scheduler (with a time-slice much shorter than the jobs' run lengths) provides better average turnaround time than FIFO. The opposite as all jobs finish at N times their times, while in FIFO some wait much less.
9. □**True** □**False:** The longer the time quantum, the more RR scheduler looks like a FIFO scheduler.
10. □**True** □**False:** Threads that are part of the same process share the same stack. Threads have own stack.
11. □**True** □**False:** With kernel-level threads, multiple threads from the same process can be scheduled on multiple CPUs simultaneously. But may need to communicate.
12. □**True** □**False:** If 3 processes with 80% wait time each are run RR for one hour then each process will get at least 12 minutes of CPU time during that hour. Wait is around (.8)**3: 50%: Total 30 mins. Each gets around 10 minutes.
13. □**True** □**False:** In producer/consumer relationship and a finite-sized circular shared buffer, producing threads must wait until there is an empty element of the buffer. Need place to produce
14. □**True** □**False:** A thread can hold only one lock at a time. Can hold as many locks as needed
15. □**True** □**False: Test_and_Set** instruction is part of hardware support for synchronization. Instruction set element: part of HW.
16. □**True** □**False:** The scheduler is the part of an OS that determines the priority of each process. Scheduler has no say in defining priorities. It may use them, though.
17. □**True** □**False:** Incrementing an integer valued variable can always be performed atomically. It may involve interruptable reading, incrementing, writing.
18. □**True** □**False:** As a rule of thumb, time quantum should be at least larger than 90% of bursts. 80
19. □**True** □**False:** No process management algorithm is better than all others for most streams. Depends on the circumstances, process mix (duration, arrivals, priorities,..)
20. □**True** □**False:** The degree of multiprogramming is the max number of jobs that can be in the running state at any given time +1. Ready+Waiting+1. Running =# of Cores (1 for single core).

Q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T	□T
□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F	□F

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

جامعة بيرزيت

BIRZEIT UNIVERSITY

Electrical and Computer Engineering Department

ENCS339 Operating Systems 1st Semester 2018/2019

Midterm Exam   Instructor: Dr. Adnan H. Yahya   Time: 90min

Student Name:__________________________ Student Number:___________

Please answer all questions using the exam sheets ONLY.

Please show all steps of your solutions. Max grade is:107.

Question 1 (18%) A computer system has only 32GB of **physical memory (RAM)**. The system has a 16KB **page size** and 48-bit logical address space. CPU generated addresses are 6 bytes each[yes: not a power of 2!].

(a) 2% 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**). Most significant (MSB) is bit #0 and least significant (LSB) is bit #47
16KB=2**14 Bytes, thus 14 bits [34:47] are used for offset (displacement) and

0	10	20	30	34 47
				++++Displacement++

(b) 2% How many **frames** are there in the RAM?

RAM is 32GB, each frame is 16KB, # of Frames= 32GB/16KB=2MFrames [addressable using 21 bits]

(c). 2% Ignoring page table overhead and OS needs, how many **pages** can a process have (max) to be runnable in **contiguous** memory allocation mode?
32GB=2MPages (2 mega pages)

(d). 2% How many **bits** are **minimally** needed for frame number of this computer in page map tables (PMTs)?
21 bits to address the 2MFrames,

(e) 2% 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)?
Flat means the table has 4GB/16KB= 1/4MPages= 256Kpages. Each page needs 21 bits or 4bytes for addresses of frames for a total of 256x4K=1MB. 1MB =1MB/16KB=220/2**14=2**6=64 pages.**

(f) 2% Given the 4GB Process: how many levels are needed for the PMT using multi-level paging of PMT, if needed?
First level has 16KB/4Bytes=4KPages=4K*16KB=64MB.
Second level has 4K*4KPages=16M*16KB=256GB.
So we have only 2 levels.

(g)3% With a **two level** paging of PMT, find the maximum size (address space, in bytes) that a job **can** have?
256GM, as shown earlier.

(h) 3% How many levels of page tables would be required to map a full 48 bit virtual address space (top level: one page max)? Explain.
2 levels gave 256GB or 238B, 3 Levels will give (2**38) x 4K=(2**38) x (2**14)=2**52Bytes; So we need 3 levels.**
Another way: Each page has 4K entries. Needs 14 bits. So 14bits for displacement, 12bits for first level, 12 for second for a total of 14+12+12=38bits. The last 10 bits are for the third level! Note that size of such job with these 3 levels (last level has only 1K entries out of 4K) is 248B= 256TB**

Question 2 (15%) Consider a computer system involving 5 processes (P1, P2, P3, P4, P5) and 4 different types of resources (R1,R2,R3,R4). The current state of the processes and resources is reflected in the tables below.

Currently Available Resources			
R1	**R2**	**R3**	**R4**
1	**4**	**2**	**0**

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

(a) 5% 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

Deadlocked □YES □ **NO**

P1 → [1,5,3,2]P3→[2,8,8,6]P2→[3,8,8,6]P4→[3,14,11,8]P5→[3,14,12,12] **Order: smallest index first**

P1 → [1,5,3,2]P4→P2→P3→P5; P1 → [1,5,3,2]P4→P2→P3→P5; ..

P4 → [1,10,5,2]P1→[1,11,6,4]P2→[2,11,6,4]P3→[3,14,11,8]P5→[3,14,12,12] **Order: P4 THEN smallest index first More exist**

If Not deadlocked: Execution Order is (just add indices): P1→ P3→ P2→ P4→ P5

2% Fill the following table:

Total Resources in the System			
R1	**R2**	**R3**	**R4**
3	**14**	**12**	**12**

(b)4% If a request from process P1 asks for the resource vector (0, 2, 0, 1).
Can the request be immediately granted? Why or why not? If yes, show an execution order. Explain your answer.

Request Can be Granted: □**YES** □ **NO**

Exceeds max of resource R4

If granted, Execution Order is (just add indices): P___→ P___→ P____→ P___→ P___

(c)4% If instead of (b), process P2 asks for the resource vector (0, 3, 2, 0), can the request be immediately granted? Why or why not? If yes, show an execution order. Explain your answer.

If (0, 3, 2, 0) is granted, Available becomes: [1,1,0,0], Still needs P2=[1,4,3,0]; Available is less than still needs for all. None can start. None can finish.

Request Can be Granted: □**YES** □ **NO If granted Available= []]]]]]]]]]]]]]]]**

P1 → [1,5,3,2]P3→[2,8,8,6]P2→[3,8,8,6]P4→[3,14,11,8]P5→[3,14,12,12] **Order: smallest index first**

No process can finish.

If granted, Execution Order is (just add indices): P___→ P___→ P____→ P___→ P___

Question 3 (16%: 4% each) Consider a dynamic (contiguous) 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):

4 → 12 → 18 → 8 → 9 → 13 → 21 → 19

↑ Hole List Start Pointer (HLPTR)

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). Assume that **the hole List Start Pointer** is moved to the closest hole to the allocated (or to the newly created after each allocation): from left to right and **circular**.

i) First Fit-5%:

4 → 1 → 18 → 8 → 9 → 13 → 21 → 19

4 → 1 → 9 → 8 → 9 → 13 → 21 → 19

ii) Worst Fit -5%:

4 → 12 → 18 → 8 → 9 → 13 → 10 → 19

4 → 12 → 18 → 8 → 9 → 13 → 10 → 10

iii) Best Fit-:

4 → **1** → 18 → 8 → 9 → 13 → 21 → 19

4 → **1** → 18 → 8 → X → 13 → 21 → 19

iv) Best Fit Plus 3- meaning best fit but each process gets exactly (size +3) hole:

4 → 12 → **4** → 8 → 9 → 13 → 21 → 19

4 → X → **4** → 8 → 9 → 13 → 21 → 19

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

Draw the Gantt charts for and compute the turnaround and wait times and fill the table entries.
F: Finish Time, TA: TurnAround Time, W: Wait Tim

Process	Arrival time	CPU burst time	Priority	Priority/P			FCFS			SJF			SRTF		
				F	TA	W	F	TA	W	F	TA	W	F	TA	W
A	3	10	1	34	31	21	34	31	21	20	17	7	20	17	7
B	14	10	1	44	30	20	44	30	20	30	16	6	30	16	6
C	0	10	5	10	10	0	10	10	0	10	10	0	10	10	0
D	2	15	5	24	22	7	24	22	7	44	42	27	44	42	27
Avge				X	23.25	12	X	23.25	12	X	21.25	10	X	21.25	10

(a) Priority/preemptive:

0

Time	910 24 34 44
Process	CCCCCC DDDDDDDDD AAAAAAAAABBBBBBBBBBB

(b) FCFS (First Come First Served).

0

Time	910 24 34 44
Process	CCCCCC DDDDDDDDD AAAAAAAAABBBBBBBBBBB

(c) SJF (Shortest Job First).

0

Time	910 20 30 44
Process	CCCCCC AAAAAAAAABBBBBBBBB DDDDDDDDD CCCCCC BBBBBBBBB AAAAAAAAA DDDDDDDDD

(d) SRTF (Shortest Remaining Time First).

0

Time	910 20 30 44
Process	CCCCCC AAAAAAAAABBBBBBBBB DDDDDDDDD CCCCCC BBBBBBBBB AAAAAAAAA DDDDDDDDD

Question 5 (18%) The producer-consumer problem is a common example of cooperating processes. A **producer process** produces information that is consumed by a **consumer process**. Here, the producer process and the consumer process communicate using a **bounded buffer** implemented in shared memory.

Producer Process | **Consumer Process**

Memory region shared by both processes:

```
#define BUFFER_SIZE 10

typedef struct {
    . . .
} item;

item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

Producer Process

```
item nextProduced;

while (1) {
   /* produce an item in nextProduced
*/
     while (((in + 1) % BUFFERSIZE) ==
out)
         ;  /* do nothing */
     buffer[in] = nextProduced;
     in = (in + 1) % BUFFER_SIZE;
}
```

Consumer Process

```
item nextConsumed;

while (1) {
     while (in == out)
         ;  /* do nothing */
     nextConsumed = buffer[out];
     out = (out + 1) % BUFFER_SIZE;
     /* consume the item in
nextConsumed */
}
```

a. (5%)Assume that the **Consumer Process** happens to be the first to run. Assume that the Consumer Process is allowed to run for a long time. **Select what happens. Explain your answer.**

1- The Consumer will be busy waiting **2- The buffer is full which produces an exception (fault).**

3- Buffer will be filled due to the long time **4- Control will be passed immediately to Producer process.**

in=out=0 and nothing can happen except busy waiting (2% for explanation)

b. (5%) Assume that the Consumer Process eventually is swapped out (or the very long time quantum is finished), and the **Producer Process** gets its chance to run. Assume that the Producer Process is allowed to run for a long time, (enough time to fill the buffer). **Select what happens. Explain your answer.**

1- The producer process will be busy waiting **2- Control will passed immediately to Producer process.**

3-Buffer will be filled due to the long time then process goes to busy waiting.

4-The buffer will overflow and an exception (interrupt) will be generated.

After the buffer is full. (in+1) % BUFFERSIZE=out and nothing can happen except busy waiting (2% for explanation)

c. (4%)For this part of the problem, assume we have re-started both processes, so they are just ready to start, with the shared memory variable having their values as initialized in the code. Describe one very fortunate **(optimistic)** sequence of executions which allows the processes to keep doing useful work. Your answer might take the form: Producer process runs until _once__ then Consumer process runs once until _OR Producer process runs until _buffer is full then Consumer process runs until Buffer is empty (or any alternating arrangement: add 2 remove 2, add 3 remove 3 and so on).

d. (4%)Is this program in need of improvement? If so, Suggest at least one way to improve performance. YES, E.g. remove busy waiting for example by sleep awake,

---

Question 6 (20%) True or false, add a line (only one) of explanation as to WHY. Fill the table (-3% if not), (-3% max for no explanations)

1. **□True □False:** An operating system is a program that acts as an intermediary between the user of a computer and the computer hardware Users cannot access hardware except through OS
2. **□True □False:** A user-level process cannot modify its own page table entries OS task
3. **□True □False:** Context switch time on modern hardware is small enough to be ignored entirely when designing a CPU scheduler. All is relative: context switch time needs to compare to time quantum
4. **□True □False:** Races happen in processes when the final result is affected by execution order.
5. **□True □False:** In a multiprocessor system with enough CPUs (cores) a process gets assigned to a given processor (core) permanently to avoid context switches. No relation. Context switch even 1 core
6. **□True □False:** Paging avoids the problem of external fragmentation of memory in a multi-programming environment but has internal fragmentation. Any frame can be used, so no external fragmentation. Can have 1 Byte or Full
7. **□True □False:** A process can move form a **ready** state to the **waiting** state, say if a device it needs becomes available. Through Running State
8. **□True □False: I**n a symmetric multiprocessor, threads cannot always be run on any processor. Symmetric means equal power/capability.
9. **□True □False:** An atomic operation is a machine instruction or a sequence of instructions that must be executed to completion without interruption. Finished in full in one go.
10. **□True □False:** Shortest Job First and Priority scheduling algorithms can lead to starvation? Short jobs keep coming all the time preventing longer jobs from being scheduled.
11. **□True □False:** Two processes reading from the same physical address access the same contents. One way of sharing
12. **□True □False:** A SJF scheduler may preempt a previously running longer job. Only after finishing previously running jobs we invoke the SJF scheduler.
13. **□True □False:** If all jobs have identical run lengths, a RR scheduler (with a time-slice much shorter than the jobs' run lengths) provides better average turnaround time than FIFO. Take much longer to finish, all jobs.
14. **□True □False:** The longer the time slice, the more RR scheduler looks like a FIFO scheduler. Most jobs will finish within one time quantum, if not all: FCFS.
15. **□True □False:** If a physical address is 32 bits and each page is 4KB, the top (Most Significant) 18 bits exactly designate the physical page number. 4KB needs 12 bits and leaves 20 not 18 for page#
16. **□True □False:** Paging approaches suffer from internal fragmentation, which decreases as the size of a page decreases. Any frame can be used, so no external fragmentation. Can have 1 Byte or Full
17. **□True □False:** Threads that are part of the same process share the same stack. Different stacks/scratchpads for different threads.
18. **□True □False:** With kernel-level threads, multiple threads from the same process can be scheduled on multiple CPUs simultaneously. Of course: parallelism is a good product of threading.
19. **□True □False:** With producer/consumer relationships and a finite-sized circular shared buffer, producing threads must wait until there is an empty element of the buffer. No writing to a full buffer: basic synchronization premise.
20. **□True □False:** A thread can hold only one lock at a time. As many. Keys are for resources. A thred can have many locks.

Q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
□T □F	**□T** □F	**□T** □F	□T **□F**	**□T** □F	□T **□F**	**□T** □F	□T **□F**	□T **□F**	**□T** □F	**□T** □F	**□T** □F	□T **□F**	□T **□F**	**□T** □F	□T **□F**	**□T** □F	□T **□F**	**□T** □F	**□T** □F	□T **□F**

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**)

0 10 20	30 40	47
[0-31: p]	[32-47: 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):

10	→	4	→	18	→	9	→	7	→	20	→	16	→	15

↑ 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.

1/9	→		→	7/11	→		→		→		→		→	

ii) Worst Fit -5%:

	→		→	9/9	→		→		→	9/11	→		→	

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

	→		→		→		→	~~0/9~~	→		→		→	4/11

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.

Q	ABET	Max	Earned
Q1	e	20	
Q2	e	20	
Q3	a	15	
Q4	c	16	
Q5	c	20	
Q6		17	
∑		108	

جامعة بيرزيت

BIRZEIT UNIVERSITY

Electrical and Computer Engineering Department

ENCS339 Operating Systems 2[ed] Semester 2017/2018

Midterm Exam Instructors: Dr. Adnan H. Yahya, Dr Ahmad Afaneh Time:90min

Student Number:________________ Student Name :________________________

Please answer **all questions** (2 sections) using the exam sheets ONLY and be BRIEF.
Please **show all steps** of your solutions. Max grade is:**108**

Question 1 (20%) A computer system has 16GB of **physical memory (RAM)**. The system has an 16KB **page size** and 32-bit logical address space. CPU generated addresses are 4 bytes each.

(a) Indicate on the diagram below which of the bits of the **logical address** of 46 bits are used for page number (**p**) and for offset (**d**) (4%)

0 10 20 30

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

d: [0-13] 14 bit , p [14 -31] 18 bit

b. How many **frames** are there in the RAM? (4%)

16GB/16KB = $2^{34}/2^{14} = 2^{20}$ Frames

c. Ignoring page table overhead and OS needs, how many **pages** can a process have (max) to be runnable in **contiguous** memory allocation mode? (3%)

Since the max number of pages is less than the max number of frames the answer is the max number of pages 2^{18}

d. What is the minimal number of bits needed for frame numbers of this computer page map tables (PMTs)? In bits_____18______, in Bytes_____3______?(3%)

e. Given a 12GB Process what is the size of the Page Map Table (PMT) in **bytes** and **pages.**
Can the table be placed in a ONE level table? Show why and why not. Show the final diagram of the page map table for such a job. (3%)

12GB/16KB =0.75 X 2^{20} PAGES

PMT size in bytes = 0.75 X 2^{20} x 3 B = 9 x 2^{18} B

PMT size in pages = 9 x 2^{18} B/ 16KB= 9 x 2^{4} =144 pages

f. TLB access time is 5% of RAM access time. RAM access time is 200ns. The TLB hit rate for paging **α** is 98%. Compute the effective access time EAT if only one level of paging for the page map table is used. What is the Max EAT possible in this system and how to achieve it? (3%)

EAT = .98(200+10) + 0.02(2x200+10) = 214 ns

Student Number:________________ Student Name :_______________________________

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**
3	**2**	**2**	**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				
P2	2	0	0	0	2	7	5	0				
P3	0	0	3	4	6	6	5	6				
P4	2	3	5	4	4	3	5	6				
P5	0	3	3	2	0	6	5	2				

(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

Deadlocked □YES □ NO

If Not deadlocked: Execution Order is: ________________________

(b)6% If a request from a 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. Explain your answer.

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

Student Number:_________________ Student Name :______________________________

Q3	**a**	**15**	
Q4	**c**	**16**	
Q5	**c**	**20**	
Q6		**17**	

Question 3 (15%)

a. Match the question with one correct answer.

Answer	Questions	#	Answers
4	Under what conditions does FIFO scheduling result in the shortest possible average response time?	1	If the job lengths are all the same, and much greater than the time slice length.
3	Under what conditions does round robin scheduling behave identically to FIFO?	2	Always when all jobs arrive at the same time.
1	Under what conditions does round robin scheduling perform poorly compared to FIFO?	3	If the job lengths are no longer than the length of the time slice.
5	Under what conditions does shortest job first perform much worse than round robin have the same order of job completion?	4	If the jobs happen to arrive in the ready queue with the shortest completion times first.
2	Under what conditions does shortest job first and shortest remaining time first perform the same?	5	This is never the case.

b. Match the question with as many correct answers as possible. Partitions are dynamic and the size of the partition is the same as the job size.

Answer	Questions		Answers
1,5	First Fit for hole selection is partitioned memory management	1	**Is the same as best fit if holes are ordered in the increasing size (largest last).**
4,5	Best Fit for hole selection is partitioned memory management	2	**Is the same as best fit if holes are ordered in the Decreasing size (largest first).**
3,5	Worst Fit for hole selection is partitioned memory management	3	**Has worst external fragmentation**
3,5	Random Fit for hole selection is partitioned memory management (selection is random)	4	**Has best external fragmentations**
6	Paging for memory management	5	**Has no internal fragmentation**
		6	**Has no external fragmentation**

c. Suppose two threads execute the following C code concurrently, accessing shared variables **a, b,** and **c**:

Initialization `int a = 4; int b = 0; int c = 0;`

Thread 1
```
if (a < 0){
     c = b - a;
} else { c = b + a; }
```

Thread 2
```
b = 10;
a = -3;
```

What are the **possible values** for **c** after both threads complete? You can assume that **reads** and **writes** of the variables are atomic, and that the order of statements **within each thread** is preserved in the code generated by the C compiler. Switching between threads can take place after any instruction.

T1 starts: c=0+4=4; T1 starts: then T2 b=10 then c=10+4=14; T1 starts: then T2 b=10, a=-3 then c=10+3=13; T2 starts: b=10, then T1 b=10, Then T2 a=-3 then c=103=7; T2 starts: b=10, a=-3, Then T1 then c=10-3=7;

Answer:**c=___4,7,13,14,-~~3~~**

What is happening here that causes this behavior: ____**Race**__________

Student Number:________________ **Student Name :**________________________________

Question 4 (16%) 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):

7 → 4 → 23 → 9 → 7 → 12 → 16 → 15

↑ Pointer to the first hole (Head_Free)

Suppose a new process P1 requiring 10 MB arrives, followed by a process P2 needing 11MB 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%:

7 → 4 → ~~23~~ / ~~13~~ / 2 → 9 → 7 → 12 → 16 → 15

ii) Worst Fit -5%:

7 → 4 → ~~23~~ / 13 → 9 → 7 → 12 → ~~16~~ / 5 → 15

iii) Best Fit-5%:

7 → 4 → 23 → 9 → 7 → ~~12~~ / 2 → 16 → ~~15~~ / 4

Student Number:________________ Student Name :________________________________

Question 5 (20%) Show the scheduling order for these processes under 4 policies: First Come First Serve (FCFS), Shortest-Remaining-Time-First (SRTF), Round-Robin (RR) with timeslice quantum = 1 and Priority, by filling in the Gantt chart with ID of the process currently running in each time quantum. *Assume that context switch overhead is 0 and that new RR processes are added to the* ***head*** *of the queue and new FCFS processes are added to the* ***tail*** *of the queue.*

For each of the algorithms: Priority, First Come First Served, RR and Shortest remaining time first compute the Finish time, TA time and Weighted Turnaround (W) time and the averages.

Note that weighted TA for a process equals TA didvided by CPU burst: **W= TA/CPU_Time**

Process ID	**Arrival time**	**CPU burst time**	**Priority**	**FCFS**			**SRTF**			**RR, slice=1**			**Priority/P**		
				F	**TA**	**W**	**F**	**TA**	**W**	**F**	**TA**	**W**	**F**	**TA**	**W**
A	0.0	2	2	**2**	**2**	**1**	**2**	**2**	**1**	**3**	**3**	**3/2= 1.5**	**2**	**2**	**1**
B	1.0	6	1	**8**	**7**	**7/6= 1.16**	**9**	**8**	**8/6= 1.33**	**13**	**12**	**12/6 =2**	**16**	**15**	**15/6 =2.5**
C	4.0	1	5	**9**	**5**	**5**	**5**	**1**	**1**	**5**	**1**	**1**	**5**	**1**	**1**
D	7.0	4	3	**13**	**6**	**6/4= 1.5**	**16**	**9**	**9/4= 2.25**	**16**	**9**	**9/4= 2.25**	**14**	**7**	**7/4= 1.75**
E	8.0	3	4	**16**	**8**	**8/3= 2.67**	**12**	**4**	**4/3= 1.33**	**15**	**7**	**7/3= 2.33**	**11**	**3**	**1**
Avg e		**16/5=3.2**			**28/5 =5.6**	**11.34/ 5=2.3**		**4.8**	**6.74= 1.35**		**32/5 =6.4**	**9.08/5 =1.82**		**28/5= 5.6**	**7.25/5 =1.45**

(a) FIFO/FCFS(First Come First Served):

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Process	**A**	**A**	**B**	**B**	**B**	**B**	**B**	**B**	**C**	**D**	**D**	**D**	**D**	**E**	**E**	**E**			

(b) SRTF (Shortest Remaining Time First).

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Process	**A**	**A**	**B**	**B**	**C**	**B**	**B**	**B**	**B**	**E**	**E**	**E**	**D**	**D**	**D**	**D**			

(c) Round Robin.

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Process	**A**	**B**	**A**	**B**	**C**	**B**	**B**	**D**	**E**	**B**	**D**	**E**	**B**	**D**	**E**	**D**			

(d) Priority (higher priority value, better)

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Process	**A**	**A**	**B**	**B**	**C**	**B**	**B**	**D**	**E**	**E**	**E**	**D**	**D**	**D**	**B**	**B**			

Student Number:________________ Student Name :_______________________________

Question 6 (17%) True or false and give ONE SENTENCE EXPLANATION. Then copy answers to the table at the end. Sample true false only, no explanations (though needed in exam).

1. **□True □False:** A Cycle in resource allocation graph indicates a deadlock when we have only one (single) instance of each resource.

2. **□True □False:** Aging is used as a mechanism to solve the deadlock problem.

3. **□True □False:** Signal() primitive usually results in resources made available while wait() usually reduces the number of available resources.

4. **□True □False:** Compaction is a costly process in memory management.

5. **□True □False:** Starvation happens when the system doesn't have the resource a process needs and thus the process cannot progress.

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

7. **□True □False:** Shortest-job-first scheduling is optimal in the sense that no other scheduling results in better throughput for a collection of processes.

8. **□True □False:** Given a constant number of bits in a virtual address, the size of a linear page table decreases with larger pages (page size).

9. **□True □False:** In real time systems the process with the earliest deadline has always to start immediately after the previous process finishes and can never be forced to wait.

10. **□True □False:** In a multiprocessor system with multiple cores a process gets assigned to a given processor (affinity) to reduce cache misses.

11. **□True □False:** Threads that are part of the same process share the same stack.

12. **□True □False:** A process can move form a **ready** state to the **waiting** state, say if it consumes all of its time quantum.

13. **□True □False:** With kernel-level threads, multiple threads from the same process can be scheduled on multiple CPUs (cores) simultaneously.

14. **□True □False: I**n a symmetric multiprocessor, threads can not always be run on any processor.

15. **□True □False:** Locks prevent the OS scheduler from performing a context switch during a critical section.

16. **□True □False:** Last Come First Served scheduling algorithm can lead to starvation.

17. **□True □False:** A Job can have several processes and a process can have several threads.

Please make sure to copy your answers to the following table (-3 points if not copied).

Question	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
□True OR □False	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F	□T □F

OS Questions

In operating system, each process has its own

a. pending alarms, signals and signal handlers

b. address space and global variables

c. all of the mentioned

d. open files

How much of its time quantum (slice) a process can use?

a. Less than one slice (Quantum) per turn

b. More than one slice (Quantum) per turn

c. All of the above

d. Exactly one slice (Quantum) per turn

Which is NOT and operating system?

a. Linux

b. Office

c. Chrome

d. Ubuntu

e. Android

What is the ready state of a process?

a. none of the mentioned

b. when process is unable to run until some task has been completed

c. when process is waiting to be scheduled for the CPU

d. when process is using the CPU

A process can be terminated due to

a. fatal error

b. all of the mentioned

c. normal exit

d. killed by another process

What is a Process Control Block?

a. A Block in memory

b. Data Structure

c. Process type variable

d. A secondary storage section

The state of a process is defined by

a. the activity just executed by the process
b) the activity just executed by the process
c) the activity to next be executed by the process
d) the current activity of the process
b) the activity just executed by the process
c) the activity to next be executed by the process
d) the current activity of the process

b. the final activity of the process

c. the activity to next be executed by the process

d. the current activity of the process

A single processor (core) system can work on more than one process during a time period

a. True

b. False

A process can NOT move from

a. Running to Ready

b. Running to New

c. Running to Waiting

d. Running to Terminated

A process can move to Ready state from:

a. New state

b. Ready state

c. Terminated state

d. Waiting State

Processes with large serial components are more likely to benefit from larger numbers of cores

a. False

b. True

Given the following fragment:

While x>2 {x=x-1};

Print x;

If x is initially = 1 then

a. The fragment will print 0

b. None of the mentioned.

c. The fragment will print 1

d. The fragment will print 2

The switching of the CPU from one process or thread to another is called

a. process switch

b. task switch

c. all of the mentioned

d. context switch

Termination of the process terminates

a. first thread of the process

b. first two threads of the process

c. no thread within the process

d. all threads within the process

Which one of the following is not shared by threads?

a. none of the mentioned

b. stack

c. both program counter and stack

d. program counter
b) stack
c) both program counter and stack
d) none of the mentioned

Thread synchronization is required because

a. None of the mentioned

b. all threads of a process share the same global variables

c. all threads of a process can share the same files

d. all threads of a process share the same address space

A single core processor can support:

a. Multithreading

b. Concurrency

c. Parallelism

d. Multiprogramming

e. All of the mentioned

Cooperating processes share many resources while independent processes share only some resources.

a. True

b. False

It is possible to have Multiple Hardware Threads even when we have a single Core machine

a. True

b. False

An I/O bound program will typically have

a. many very short CPU bursts

b. many very short I/O bursts

c. a few very short CPU bursts

d. a few very short I/O bursts

Consider three CPU-intensive processes, which require 10, 20 and 30 time units and arrive at times 0, 2 and 6, respectively. How many context switches are needed if the operating system implements a shortest remaining time first scheduling algorithm? Do not count the context switches at time zero and at the end.

a. 1

b. None of the mentioned

c. 2

d. 3

e. 4

A scheduling algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (the lowest priority). The scheduler re-evaluates the process priorities every T time units and decides the next process to schedule. Which one of the following is TRUE if the processes have no I/O operations and all arrive at time zero?

a. This algorithm is equivalent to the first-come-first-serve algorithm

b. This algorithm is equivalent to the shortest-job-first algorithm.

c. This algorithm is equivalent to the round-robin algorithm.

d. This algorithm is equivalent to the shortest-remaining-time-first algorithm

Assume every process requires 3 seconds of service time in a system with single processor. If new processes are arriving at the rate of 10 processes per minute, then estimate the fraction of time CPU is busy in system?

a. 50

b. 30

c. 60

d. 20

Consider three processes (process id 0, 1, 2 respectively) with compute time bursts 2, 4 and 8 time units. All processes arrive at time zero. Consider the longest remaining time first (LRTF) scheduling algorithm. In LRTF ties are broken by giving priority to the process with the lowest process id. The average turn around time is:

a. **13 units**
b. None of the mentioned
c. 16 units
d. 14 units
e. 15 units

In which of the following scheduling criteria, context switching will never take place back to the same process?

a. Preemptive priority
b. Preemptive SJF
c. ROUND ROBIN
d. **Non-preemptive SJF**

Which of the following statements is not true for Multi Level Feedback Queue processor scheduling algorithm?

a. Each queue may have different scheduling algorithm
b. **Processes are permanently assigned to a queue**
c. Queues have different priorities
d. This algorithm can be configured to match a specific system under design

Which of the following scheduling algorithms may cause starvation ?

a. First-come-first-served

b. Round Robin

c. Priority

d. Shortest process next

e. Shortest remaining time first

Select one:

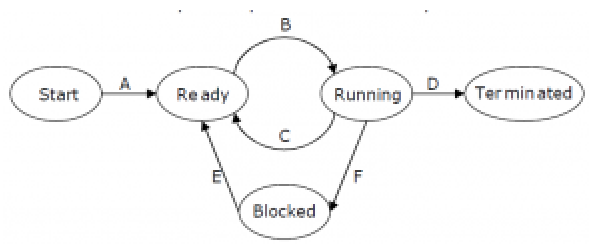
a. **c, d and e**

b. b, c and d

c. b, d and e

d. a, c and e

In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state: Now consider the following statements:



I. If a process makes a transition D, it would result in another process making transition A immediately.

II. A process P2 in blocked state can make transition E while another process P1 is in running state.

III. The OS uses preemptive scheduling.

IV. The OS uses non-preemptive scheduling.

Which of the above statements are TRUE?

a. I and II

b. I and III

c. **II and III**

d. II and IV

BIRZEIT UNIVERSITY
DEPARTMENT OF COMPUTER SYSTEM ENGINEERING

ENCS339: Operating Systems — Second Semester 2010/2011

Mr. Abdel Salam Sayyad — **Second Exam** — Date: 27/4/2011

Student Name: **Typical Solutions** **Student Number: 0000000**

Question 1: (25 marks)

The Sleeping-Barber Problem: A barbershop consists of a waiting room with n chairs and a barber room with one barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers (i.e. write the barber process and the customer process).

Solution:

Need to define the following variables with the indicated initial values:

```
Semaphore Customers = 0 //initially there are no customers
Semaphore Barber (mutex) = 0 //initially the barber is asleep
Semaphore accessSeats (mutex) = 1 //aquire this lock before changing
                              // the Number of Free Seats
int NumberOfFreeSeats = n // initially all seats are free
```

The barber process:

```
while(true) {
   wait(Customers);     //tries to acquire a customer
                        //if none is available he goes to sleep
   wait(accessSeats);   //the barber is now awake
                        // acquire the lock on free seats
   NumberOfFreeSeats++; //one seat gets free
   signal(Barber);      //the barber is ready to cut hair
   signal(accessSeats); //release the lock on free seats
   //here the barber is cutting hair
 }
```

Continued Next Page

More Space for Question 1

The customer process:

```
while(true) {
   wait(accessSeats);   // acquire the lock on free seats
   if ( NumberOfFreeSeats > 0 ) { //if there are any free seats
     NumberOfFreeSeats--; //sitting down on a seat
     signal(Customers); //notify the barber that there's a customer
     signal(accessSeats); // release the lock on free seats
     wait(Barber);        //now it's this customer's turn,
                          //but wait if the barber is busy
     //here the customer is having his hair cut
   } else {               //there are no free seats - go home
     signal(accessSeats); //release the lock on the seats
   }
 }
```

<u>Question 2</u>: (25 marks)

Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a) What is the content of the Need matrix?
b) Is the system in a safe state?
c) If a request from process P1 arrives for (0,4,2,0), can the request be granted immediately?

<u>Solution:</u>

a) Need = Max – Available = { 0 0 0 0 , 0 7 5 0 , 1 0 0 2 , 0 0 2 0 , 0 6 4 2 }

b) Banker's algorithm:

Initially: Work = Available = { 1 5 2 0 }, finish = { *false false false false false* }

Iteration 1: for P0, Need < Work, Work = { 1 5 2 0 } + { 0 0 1 2 } = { 1 5 3 2 }, finish[0] = *true*

Iteration 2: for P2, Need < Work, Work = { 1 5 3 2 } + { 1 3 5 4 } = { 2 8 8 6 }, finish[2] = *true*

Iteration 3: for P3, Need < Work, Work = { 2 8 8 6 } + { 0 6 3 2 } = { 2 14 11 8 }, finish[3] = *true*

Iteration 4: for P4, Need < Work, Work = { 2 14 11 8 } + { 0 0 1 4 } = { 2 14 12 12 }, finish[4] = *true*

Iteration 5: for P1, Need < Work, Work = { 2 14 12 12 } + { 1 0 0 0 } = { 3 14 12 12 }, finish[1] = *true*

Therefore, the system is in a safe state, and a safe sequence is P0, P2, P3, P4, P1.

<u>Continued Next Page:</u>

More Space for Question 2

c) First we notice that the new request { 0 4 2 0 } is less than available resources { 1 5 2 0 }, and is also less than Need_1 { 0 7 5 0 }, and so we proceed to pretend that we granted the request and see if the new state is safe:

The new state:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	1 1 0 0
P1	1 4 2 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

Need = Max – Available = { 0 0 0 0 , 0 3 3 0 , 1 0 0 2 , 0 0 2 0 , 0 6 4 2 }

Banker's algorithm:

Initially: Work = Available = { 1 1 0 0 }, finish = { *false false false false false* }

Iteration 1: for P0, Need_0 < Work, Work = { 1 1 0 0 } + { 0 0 1 2 } = { 1 1 1 2 }, finish[0] = *true*

Iteration 2: for P2, Need_2 < Work, Work = { 1 1 1 2 } + { 1 3 5 4 } = { 2 4 6 6 }, finish[2] = *true*

Iteration 3: for P3, Need_3 < Work, Work = { 2 4 6 6 } + { 0 6 3 2 } = { 2 10 9 8 }, finish[3] = *true*

Iteration 4: for P4, Need_4 < Work, Work = { 2 10 9 8 } + { 0 0 1 4 } = { 2 10 10 12 }, finish[4] = *true*

Iteration 5: for P1, Need_1 < Work, Work = { 2 10 10 12 } + { 1 4 2 0 } = { 3 14 12 12 }, finish[1] = *true*

Therefore, the system is in a safe state, and a safe sequence is P0, P2, P3, P4, P1.

Thus the new request can be granted.

<u>Question 3</u>: (20 marks)

Consider a demand-paging system with the following time-measured utilizations:

CPU utilization: 20%
Paging disk utilization: 97.7%
Other I/O devices utilization: 5%

Which (if any) of the following will (probably) improve CPU utilization? Explain your answer.

<u>Solution</u>: The system obviously is spending most of its time paging, indicating over-allocation of memory. If the level of multiprogramming is reduced resident processes would page fault less frequently and the CPU utilization would improve. Another way to improve performance would be to get more physical memory or a faster paging disk.

a) Install a faster CPU.

No , because this will not reduce page fault rate.

b) Install a bigger paging disk.

No , because this will not reduce page fault rate.

c) Decrease the degree of multiprogramming.

Yes , because this will reduce page fault rate by allowing more pages of the same process to be resident in memory.

d) Install more main memory.

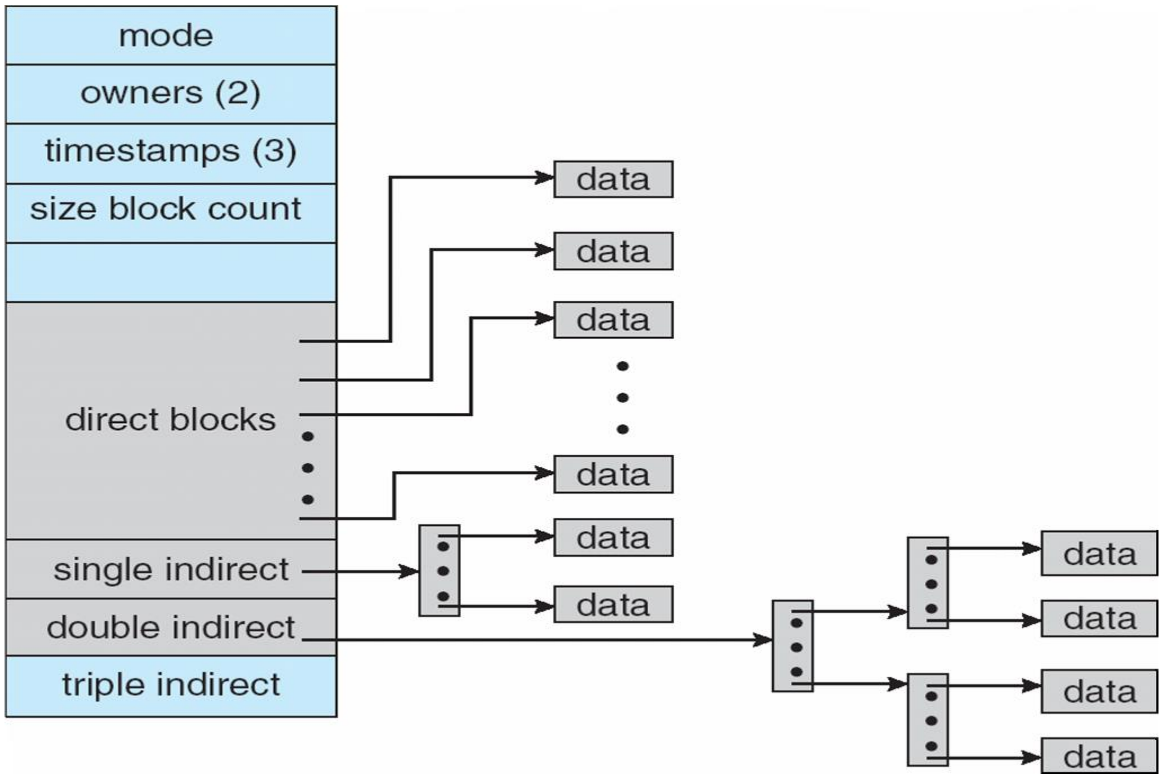
Yes , because this will reduce page fault rate by allowing more pages of the same process to be resident in memory.

e) Increase the page size.

Increasing the page size will result in fewer page faults if data is being accessed sequentially. If data access is more or less random, more paging action could result because fewer pages can be kept in memory and more data is transferred per page fault. <u>So this change is as likely to decrease utilization as it is to increase it.</u>

Question 4: (30 marks)

Consider the UNIX file system with the inode shown:



Knowing that the number of direct block pointers is 12, the block size is 4K bytes, and the block pointer size is 32 bits, answer the following:

a) Assuming that we won't use the triple indirect block pointer, what would be the maximum allowable file size? **[5 marks]**

Solution:

The 12 direct blocks can house 12 * 4 KB = 48 KB

The single indirect block has 4 KB / 4 = 1024 pointers, each pointing to a 4 KB block, amounting to 1024 * 4 KB = 4096 KB

The amount pointed at by the double indirect blocks is 1024 * 1024 * 4 KB = 4194304 KB

The total file size is 48 + 4096 + 4194304 = 4198448 KB

b) If the O.S. wants to allocate space for a 70K file, and the list of free blocks is:

5, 6, 3, 4, 8, 11, 12, 15, 16, 7, 9, 13, 14, 17, 18, 19, 20, 24, 25, 22, 23, 26, 27…

Show the contents of the block pointers in the inode. **[10 marks]**

Solution:

We need 18 blocks to house the data (18 * 4 KB = 72 KB),

Pointer #	Type	Contents
1	direct	5
2	direct	6
3	direct	3
4	direct	4
5	direct	8
6	direct	11
7	direct	12
8	direct	15
9	direct	16
10	direct	7
11	direct	9
12	direct	13
13	single indirect	14
14	double indirect	-1
15	triple indirect	-1

Block #14 has pointers to six other blocks as follows:

Pointer #	Contents
1	17
2	18
3	19
4	20
5	24
6	25
7	-1
8	-1
9	-1
	. . . .
1024	-1

c) What is the amount of internal fragmentation in b? **[5 marks]**

Solution:

There are two blocks with internal fragmentation:
The single indirect block has 6 pointers only, 6 * 4 = 24 Bytes, which leaves 4096 – 24 = 4072 Bytes of internal fragmentation.
The last data block has 2 KB of internal fragmentation.

The total internal fragmentation is 4072 + 2048 = 6120 Bytes.

d) If the file system is a DOS File Allocation Table, show the contents of the FAT for the same 70K file, and the same list of free blocks:

5, 6, 3, 4, 8, 11, 12, 15, 16, 7, 9, 13, 14, 17, 18, 19, 20, 24, 25, 22, 23, 26, 27… **[10 marks]**

Solution:

The FAT looks like this:

Entry	Contents	Entry	Contents
1		16	7
2		17	18
3	4	18	19
4	8	19	20
5	6	20	24
6	3	21	
7	9	22	
8	11	23	
9	13	24	EOF
10		25	
11	12	26	
12	15	27	
13	14	28	
14	17	29	
15	16	30	