1	
2	
3	
4	
5	
6	
total	

Name Key

1. [16 points total] Consider the following snapshot of a system:

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

Answer the following questions using the banker's algorithm:

a. [4 points] Show the content of the matrix *Need* below.

Need				
	A	B	C	D
P0	2	2	1	1
P1	3	0	1	0
P2	0	3	2	3
P3	0	1	1	0
P4	2	3	0	3

b. [4 points] Show that the system in a safe state by listing the order in which processes can be executed without producing a deadlock.

<P3, P0, P4, P2, P1>

c. [8 points] If a request from process P_2 arrives for (0, 0, 0, 1), can the request be granted immediately? If yes, list the order of process execution. **Show the updated NEED and Allocation matrices as well as Availabl** ***If yes, show the safe sequence. If no, list the processes that are possibly in a deadlock.***

	NEED	ALLOC	AVAIL
P0	2211	0101	0110
P1	3010	0201	
P2	0322	1002	
P3	0110	2201	
P4	2303	0020	

NO, P1, P2 & P4 are possibly in deadlock

2. [24 points total] Consider the following page frame tables and reference string. There are a total of 4 page frames available. Fill the page frames starting from the topmost frames (***points will be deducted for starting anywhere else***).

a. Construct the page replacement table for FIFO algorithm and ***record how many page faults***. **Fill in the page table after each reference.**

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

11 page fault

b. Construct the page replacement table for LRU algorithm and ***record how many page faults***. **Fill in the page table after each reference.**

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

13 page faults

c. Construct the page replacement table for 2nd chance algorithm and ***record how many page faults***. **Fill the page tal after each reference.**

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

12 page faults

d. Construct the page replacement table for MFU algorithm and ***record how many page faults***. **Show the page table after each reference. Tie-breaker: choose topmost frame**

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

16 page faults

3. [36 points] Consider a machine with a frame size of 512KB.

a. If we want a virtual memory of 16 Petabytes, how many nonzero bits are needed in the virtual address?

54 bits (note 16PB = 2^{54} bytes)

b. Maximally how many nonzero bits would be needed for the index into a single level page table?

35 bits |←35 → ← 19 →| |← 54 →|

c. If frame IDs in this machine are 32 bits long, how much physical memory could this machine have maximally?

$2^{32+19} = 2^{51}$ bytes

d. How many bits are needed for page displacement?

19 bits (note page size is 2^{19} bytes)

e. If we use a two-level page table and entries in the outer page table are 8 bytes long, how many pages in size sho the outer page table be in order to support a maximum of 128K entries?

$\frac{2^{17} \cdot 2^{3}}{2^{19}} = 2$ pages

f. How many bits are needed for an outer index for part e)? 17 bits ($128K = 2^{17}$)

g. How many bits are then used for the inner page table index? 18 bits $(35-17=18)$

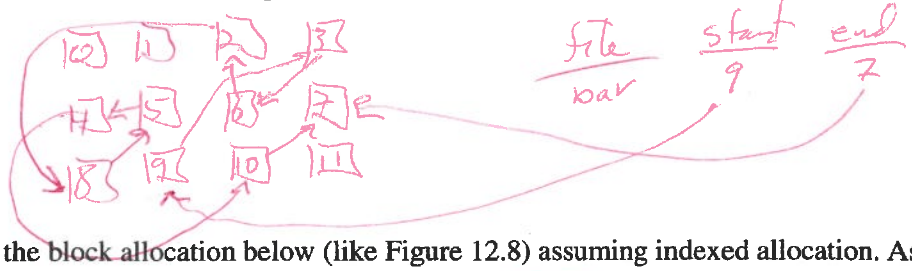
h. How large maximally is an individual inner page table in terms of 512KB pages? $\frac{2^{18} \cdot 2^{2}}{2^{19}} = 2$ pages

i. If the swap file is on a disk that uses **32KB** blocks, how many blocks maximally can a single process occupy wi the virtual address space of this machine?

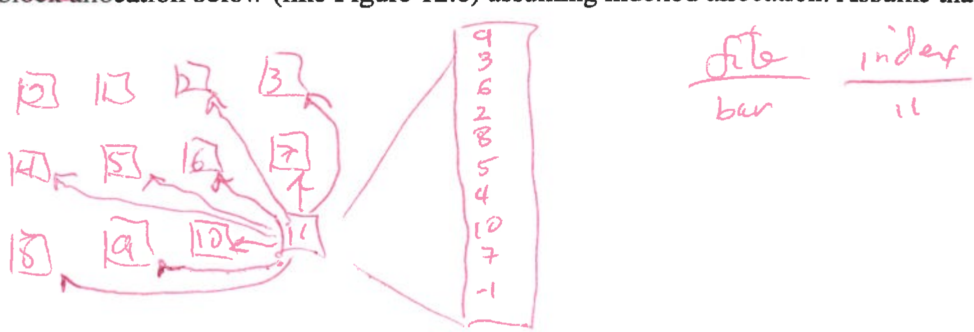
2^{39} blocks $\left(\frac{2^{54}}{2^{15}} = 2^{39}\right)$

4. [4 points] Assume the file *bar* is allocated blocks in the following order: 9,3,6,2,8,5,4,10,7.

a. Show the block allocation figure below (like Figure 12.6) assuming linked allocation.



b. Show the block allocation below (like Figure 12.8) assuming indexed allocation. Assume that block 11 is the in block.



5. [20 points] Consider a version of the bounded buffer problem in which there are one producer process (P_1) and one consumer processes(P_2 and P_3) all sharing the same buffer. Assume that the size of the buffer is n = 4, and that we start with a completely empty buffer. The structure of P_1, P_2, and P_3 as well as the semaphores and buffer is shown below:

```
/* structure of P1 ~~& P2~~ */
do {
   ......
   produce an item in nextp
   ......
   wait(empty)
   wait(mutex)
   buffer[in]= nextp
   in = (in + 1) % n
   signal(mutex)
   signal(full)
}while(1)
```

```
/* structure of P2+, P3 */
do {
   wait(full)
   wait(mutex)
   nextp = buffer[out]
   out = (out + 1) % n
   signal(mutex)
   signal(empty)
   ..........
   consume item in nextp
   ........
}while(1)
```

	item 0
	item 1
	item 2
	item3

0	in
0	out
4	empty
0	full
1	mutex

Assume multi-level queue scheduling with P2 & P3 in the high priority queue using RR scheduling in which it takes one quantum to get through the critical section code and a second quantum to consume an item. P1 is in the low priority que which uses FCFS scheduling. Processes in the high priority queue preempt processes in the low priority queue. Assume that the processes start in the MLQ at the same time. In the high priority ready queue in the order from head to tail are P3 P2, and in the low priority ready queue is P1. Assume that the semaphore queues use a priority scheme in which P2 (highest priority) > P1 > P3 (lowest priority).

Draw the contents of the indices "in" and "out", as well as the state of the semaphores and the contents of the buffer ***afte items have been consumed***. In the case of the buffers, simply notate each item with the name of the process that accesse it last.

RR → P3 → P2
FCFS → P1

P3: Full: -1 P3 blocks
P2: Full: -2 P2 blocks
P1: produce item, MT: 3, mux: 0, item0: P1, in: 1, mux: 1, Full: -1 unblocks P2
P2: mux: 0, item0: P2, out: 1, mux: 1, MT: 4 quantum ends
P2: consumes item quantum ends
P2: Full: -1 P2 blocks
P1: produce item, MT: 3, mux: 0 item1: P1, in: 2, mux: 1, Full: -1 unblocks P2
P2: mux: 0, item1: P2, out: 2, mux: 1, MT: 4 quantum ends
P2 consumes item

P2	item 0
P2	item 1
	item 2
	item3

2	in
2	out
4	empty
-1	full
1	mutex