1	
2	
3	
4	
5	
total	

Name Key

1. [20 points] Consider the Following set of processes:

Process	Burst Time	Arrival Time
P0	22	9
P1	26	6
P2	14	3
P3	18	0

a. Draw a Gantt chart to show the SJF scheduling for these processes.

P3	P2	P0	P1	
0	18	32	54	80

b. Draw a Gantt chart to show the SRTF scheduling for these processes.

P3	P2	P3	P0	P1	
0	3	17	32	54	80

c. Draw a chart to show the RR scheduling for these processes if the quantum is 5.

P3	P2	P3	P1	P0	P2	P3	P1	P0	P2	P3	P1	P0	P1	P0	P1	P0	P1	
0	5	10	15	20	25	30	35	40	45	49	52	57	62	67	72	77	79	80

d. Assuming process priority P0:1, P1:3, P2:0, P3:2, draw a chart to show the preemptive priority scheduling for these processes.

P3	P2	P0	P3	P1	
0	3	17	39	54	80

2. [10 points] What is the average waiting time for each of the scheduling algorithms in question 1?

P0 P1 P2 P3

a [(54 - 9 - 22) + (80 - 6 - 26) + (32 - 3 - 14) + (18 - 0 - 18)]/4

b [(54 - 9 - 22) + (80 - 6 - 26) + (17 - 3 - 14) + (32 - 0 - 18)]/4

c) [(79 - 9 - 22) + (80 - 6 - 26) + (49 - 3 - 14) + (52 - 0 - 18)]/4

d) [(39 - 9 - 22) + (80 - 6 - 26) + (17 - 3 - 14) + (54 - 0 - 18)]/4

3. [25 points] Consider the following snapshot of a system:

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

Answer the following questions using the banker's algorithm:

a. Show the content of the matrix *Need* below.

	Need			
	A	B	C	D
P0	1	1	2	1
P1	2	4	2	6
P2	0	4	1	2
P3	2	1	2	0
P4	1	0	1	3

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

<P0, P2, P4, P3, P1>

c. Starting with the need matrix from (a), if a request from process P_4 arrives for (0, 0, 1, 0), can the request be granted immediately? **Show the updated NEED and Allocation matrices as well as Available. *If yes, show the safe sequence. If no, list the processes that are possibly in a deadlock.***

	NEED ABCD	ALLOC ABCD	AVAIL ABCD
P0	1121	0201	1221
P1	2426	0011	- 0010
P2	0412	0111	1211
P3	2120	0004	
P4	1003	2120	

No All processes are possibly in deadlock

d. Starting with the need matrix from (a), if a request from process P_2 arrives for (1, 0, 1, 1), can the request be granted immediately? **Show the updated NEED and Allocation matrices as well as Availability.** ***If yes, show the safe sequence. If no, list the processes that are possibly in a deadlock.***

	NEED	ALLOC	AVAIL
P0	1121	0201	1221
P1	2426	0011	-1011
P2	-1401	1122	0211
P3	2120	0004	
P4	1013	2110	

No! the request exceeds P2's Need for type "A" resources. This invalid request must be refused.

4. [15 points] Consider two counting semaphores S and T. The semaphore variable for S is initialized to 1 and the semaphore variable for T is initialized to 0 in a system that uses FCFS scheduling. Suppose that these two semaphores appear in two sections of code, A and B, but do not appear in any other sections of code.

```
/* section A */
    wait(S);
    wait(T);
    <calculate the secret of life>
    wait(S);
    return;
```

```
/* section B */
signal(T);
signal(S);
<calculate the joy in life>
signal(T);
return;
```

a) Assume process P_1 wants to execute section A and process P_2 wants to execute section B, and that no other processes want to execute section A or B. Are there any circumstances in which either or both these two processes can become deadlocked? If yes, show the order of execution that causes deadlock. If no, show that no order of execution causes deadlock.

P1 → P2
P1: S:0, T:-1 ⇒ P1 blocks
P2: T:0, unblocks P1, S:1, T:1
P1: S:0
no deadlock

P2 → P1
P2: T:1, S:2, T:2
P1: S:1, T:1, S:0
no deadlock

b) If instead of there only being two processes wanting to execute these sections, imagine we start with P_1, P_2, P_3, P_4, P_5 in which P_1 is at the head of the ready queue and P_5 is at tail. The processes P_2, P_3 and P_5 want to execute section B, while processes P_1 and P_4 want to execute section A. What is the state of the semaphores and semaphore queues after all processes have been dispatched exactly once, *i.e.*, what is the value of the semaphore variables and what if any processes are blocked on the semaphore queues?

S.value = 1
S.head→ null

T.value = 4
T.head→ null

5. [30 points] Consider a version of the bounded buffer problem in which there are two producer processes (P_1 and P_2) and one consumer process (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 and 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 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 a FCFS scheduler and that all processes start in the ready queue at the same time in the order from head to tail, P_3, P_2, and P_1 (P_3 at the head of the queue). Assume that the semaphore queues use a priority scheme in which P_1 (highest priority) > P_2 > P_3 (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 ***after 2 items have been consumed***. In the case of the buffers, simply notate each item with the name of the process that accessed it last. RQ → P3, P2, P1

P3: Full: -1 ⇒ P3 blocks

P2: produce item, MT: 3, mux: 0, item0: P2, in: 1, mux: 1, Full: 0 ⇒ unblocks P3

P2: produce item, MT: 2, mux: 0, item1: P2, in: 2, mux: 1, Full: 1

P2: produce item, MT: 1, mux: 0, item2: P2, in: 3, mux: 1, Full: 2

P2: produce item, MT: 0, mux: 0, item3: P2, in: 0, mux: 1, Full: 3

P2: produce item MT: -1 P2 blocks

P1: produce item MT: -2 P1 blocks

P3: ~~[illegible]~~ mux: 0 item0: P3, out: 1, mux: 1, MT: -1 (unblocks P1) consume item

P3: Full: 2, mux: 0, item1: P3, out: 2, mux: 1, MT: 0 (unblocks P2) Consume item

Done!

~~P2~~ P3	item 0
~~P2~~ P3	item 1
P2	item 2
P2	item3

0	in
2	out
0	empty
2	full
1	mutex