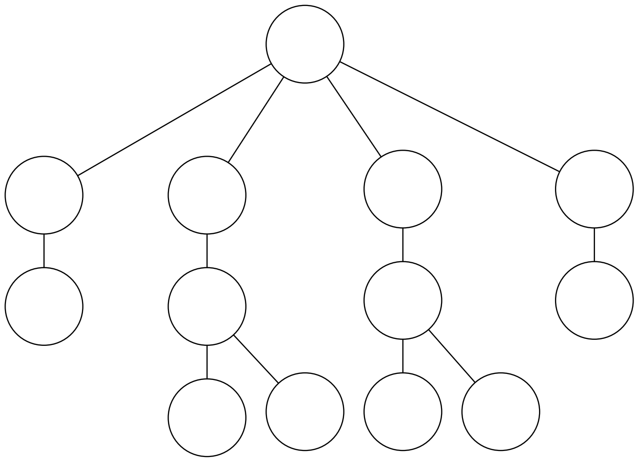
13 July 2020

Computer Operating Systems
2019-2020 Spring
Final Exam

Prof. Nadia Erdoğan, Assist. Prof. Gökhan Seçinti

Name:	
Student ID:	

Q1. (15points) Write the C code, which generates the processes hierarchy given on the right.



```
int f = 1
int i;
for(i=0; i<4; i++){
	if(f > 0)
		f=fork();
	else
		break;
}
if(f==0){	// level 1 child
	f = fork();
	if (f == 0 && (i==1 || i==2)){	// level 2 child
		for(int j=0; j<2; j++){
			int f2 = fork();
			if(f2==0)		// level 3 child
				break;
		}
	}
}
```

Q2. (20points)

a) A software-based solution is given below for the Critical Section (CS) problem. Please explain whether this solution satisfies the requirements of proper solution/implementation. Why?

Process P_i

```
do {
        turn = j;
        flag [i] = TRUE;
        while (flag [j] and turn == j);
                critical section
        flag [i] = FALSE;
                remainder section
} while (TRUE);
```

Answer:

Yes, this is a sufficient and correct software-based solution. This algorithm is also known as Peterson algorithm.

Because, before a process ***i*** enters its CS, it checks for other process ***j***, if ***j*** is in their CS, ***i*** waits (waits with the while statement). Also, if ***i*** is in its CS, ***j*** cannot enter its CS. This solution provides correct implementation for mutual exclusion.

b) An auction system will allow **n** number of processes (Pi, i=1..n) to place a bid for a resource and allocates the resource to the highest bidder. In order to place a bid, a process will call "place-bid" function with two parameters: *i) offered price ii) process's own index.* Once a proces placed a bid, it will be blocked inside the "place-bid" function until the final decision has been made, in order to prevent multiple bids from a single process. An auctioneer process checks whether **n** bids have been placed. After all bids are in place, it will pick the hightest offer and write the related price and index of the bidder process to a shared memory space and, it will release locks blocking the bidder processes.

Write the psuedo-code for the the mechanism described above including "place-bid" function and explain the use of the semaphores by auctioneer and bidder processes.

Answer:

```
BidderProcess_i{
        While(True) place-bid(offer, i);}
AuctioneerProcess{
        semaphore = 0
        wait(semaphore, n);
        (max_bid, max_i) = findMaxBid(bids_array);
        *shared_max = max_bid; *(shared_max+1) = max_i;
        signal(semaphore, n);
        sem_1=1; sem_2=1; … sem_n=1; }
place-bid(bid, i){
        wait(sem_i, 1);
        bids[i] = bid;}
// Auctioneer uses a semaphore to check if everybody has a bid, then finds max and writes it, then again allows the others to bid.
```

13 July 2020

Q3. **(15points)** Please answer the following questions, considering the following snapshot of a computer system with four (4) types of resources and five (5) active proceses.

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

i. Illustrate that the system in a safe state by showing an order in which the processes may complete.

Remaining Requests Matrix:

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

System is in a safe state. Example order: **P0, P3, P1, P2, P4**

ii. If a request from process P1 arrives for (1,1,0,0), should this request be granted immediately?

Assume this request is granted. Free resource vector is now: (2,2,2,1)

Then all processes can still finish their job with these resources. Example: P0, P3, P1, P2, P4

System is safe. The request should be granted. (YES)

iii. If a request from process P4 arrives for (0,0,2,0), should this request be granted immediately?

If this request is granted, free vector: (3,3,0,1)

In this condition, there is a deadlock possiblity, system is not safe. Request shhould not be granted. (NO.)

13 July 2020

Q4. **(20points)** In a computer system, processes A and B are CPU-bound processes (represented by “C”), while process C requires 1 time unit of processing time, followed by 4 units of input/output time, I/O, (represented by “I”).

For each of the following scheduling policies, determine

a) The process schedule,
b) The turnaround time for each process, and
c) The total number of context switching required.

To specify the schedule, use the identity of process (A,B,C) and, in case the processor is idle, use “x” . The FIFO scheduling policy is given below as an example.

Process Name	Arrival Time	Execution Pattern	Execution Time
A	0	CCCCCCCC……	10
B	1	CCCCCCCC……	11
C	2	CIIIICIIIICIIII……	16

i) **FIFO:**
a) AAAAAAAAAABBBBBBBBBBBCxxxxCxxxxCxxxxC
b) Turnaround time: A = 10 ; B= 20 ; C= 35
c) Number of context switches: 2 (A->B, B->C)

ii) Round Robin - Q=5
a) AAAAABBBBBCxxxxAAAAABBBBBCxxxxBCxxxxC
b) Turnaround time: A = 20 ; B= 30 ; C= 35
c) Number of context switches: 7

iii) Round Robin - Q=1
a) ABCABxABCABxABCABxABCABABABB
b) Turnaround time: A = 26 ; B= 27 ; C= 19
c) Number of context switches: 26

iv) Shortest Remaining Processing Time
a) AAAAAAAAAABBBBBBBBBBBCxxxxCxxxxCxxxxC
b) Turnaround time: A = 10 ; B= 20 ; C=35
c) Number of context switches: 2

v) For the given work load, which scheduling policy has made best use of processor time?

Round-Robin with 1 quantum gives the best result, because it reduces IO wait time (reduces idle CPU time)

vi) Which of the above policies is most suited for interactive processes?

Round-Robin with 1 quantum gives the best result, because it reduces IO wait time (reduces idle CPU time)

13 July 2020

Q5. **(15points)** There are free memory partitions of 100KB, 400 KB, 200KB, and 500 KB, in the given order. Where would the following memory allocation algorithms place memory requests of 200KB, 396 KB, 100 KB, and 280 KB, arriving in the given order?

a) First Fit

200 KB → In 400KB space
396 KB → In 500KB space
100 KB → In 100KB space
280 KB → cannot be placed.

b) Next Fit

200 KB → In 400KB space
396 KB → In 500KB space
100 KB → In 100KB space (500-396 = 104 KB space cannot be used because of internal fragmentation.)
280 KB → cannot be placed.

c) Best Fit

200 KB → In 200KB space
396 KB → In 400KB space
100 KB → In 100KB space
280 KB → In 500KB space

13 July 2020

Q6. (15points) Suppose there are 16 virtual pages and 4 page frames and the all page frames are initially empty. References to pages 0 1 2 3 2 4 1 3 5 6 1 3 2 7 4 8 occur in the given order.
Determine the **number of page faults** that occur for the following page replacement algorithms by showing the active content of the page frames.

a) FIFO

	0	1	2	3	2	4	1	3	5	6	1	3	2	7	4	8
Page 1	0	0	0	0	0	4	4	4	4	4	4	3	3	3	3	8
Page 2		1	1	1	1	1	1	1	5	5	5	5	2	2	2	2
Page 3			2	2	2	2	2	2	2	6	6	6	6	7	7	7
Page 4				3	3	3	3	3	3	3	1	1	1	1	4	4
p. Fault	X	X	X	X		X			X	X	X	X	X	X	X	X

There are 13 page faults.

b) LRU

	0	1	2	3	2	4	1	3	5	6	1	3	2	7	4	8
Page 1	0	0	0	0	0	4	4	4	4	6	6	6	6	7	7	7
Page 2		1	1	1	1	1	1	1	1	1	1	1	1	1	4	4
Page 3			2	2	2	2	2	2	5	5	5	5	2	2	2	2
Page 4				3	3	3	3	3	3	3	3	3	3	3	3	8
p. Fault	X	X	X	X		X			X	X			X	X	X	X

There are 11 page faults.