CSE321: Operating Systems
Quiz 1
Set A

Name:	ID:	Section:

Marks: 10 **Time: 15 minutes**

Question 1: [CO1] In a particular scenario, There are 7 processes in the ready queue for execution. All processes require almost full allocation of CPU during their execution. Explain logically what issue can arise here and what is the solution. **[Marks: 2]**

→ Empty Waiting queue.

Soln:
Long / Mid term scheduler should choose wisely.

Question 2: [CO1] At a certain moment, you are using a browser. From the browser, you have opened 3 separate tabs consisting of outlook mail, youtube and google scholar. After looking at some articles in the google scholar you have closed the tab consisting of google scholar. Explain logically by relating with the scenario what type of process termination was done here. **[Marks: 2]**

→ Task assigned to the process has been finished. Therefore it was no longer needed.

Question 3: [CO1] Show outputs of the given code. **[Marks: 6]**

```
int main(){
        pid_t a,b;
        a=fork();
        if (a<0){
                printf("Fork failed\n");
        }
        else if (a==0){
                printf("Child started\n");
                b=fork();

                if(b<0){
                        printf("Fork failed\n");
                }
                else if (b>0){
                        printf("Child started\n");
                        wait(NULL);
                        printf("Grand Child finished\n");
                        printf("Child finished\n");
                }
                else{
                        printf("Grand Child started\n");
                }

        }
        else{
                printf("Parent started\n");
                printf("Parent will be interrupted\n");
                wait(NULL);
                printf("Child finished\n");
                printf("Parent finished\n");
        }
        printf("Terminating\n");
        return 0;
}
```

11	21	31
P_1	P_2	P_3
a = 11/21	a = 0 b = 21/31	a = 0 b = 0

Outputs:

parent started
parent will be interrupted
child started
child started
Grand Child started
Terminating
Grand child finished
Child finished
Terminating
Child finished
→ Parent finished
Terminating

CSE321: Operating Systems
Quiz 1
Set B

Name:	ID:	Section:

Marks: 10 **Time: 15 minutes**

Question 1: [CO1] In a particular scenario, There are 5 processes in the ready queue for execution. All processes require almost full allocation of IO resources during their execution. Explain logically what issue can arise here and what is the solution. **[Marks: 2]**

→ Empty ready queue.

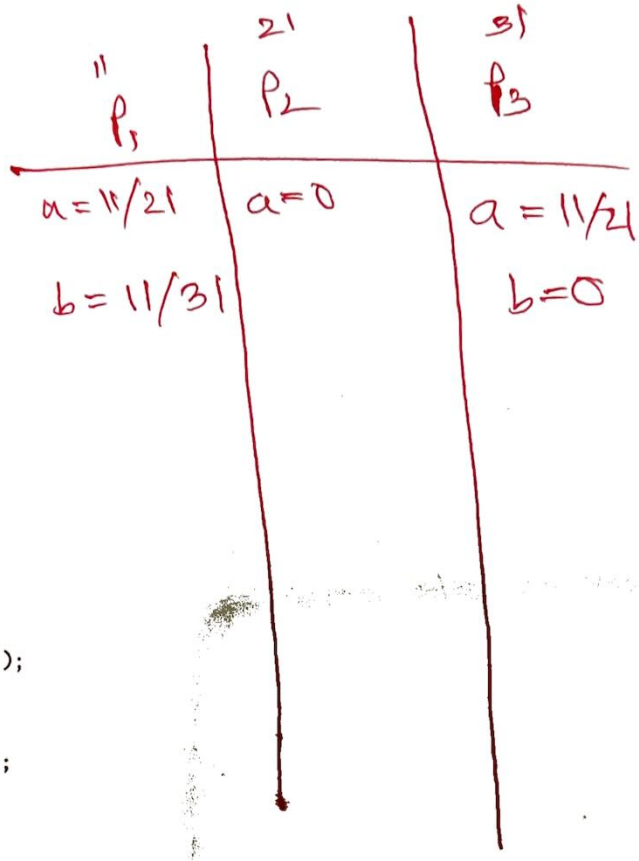
Soln:
Long / Mid term scheduler should choose wisely.

Question 2: [CO1] At a certain moment, you are using a browser. From the browser, you have opened 3 separate tabs consisting of outlook mail, youtube and a third party online IDE. After deriving some instructions in the tab consisting of the IDE when you tried to compile those the IDE required a huge space from the main memory. As a result, the browser forcefully killed that tab. Explain logically by relating with the scenario what type of process termination was done here. **[Marks: 2]**

→ If exceeded the usage of its allocated memory.

Question 3: [CO1] Show outputs of the given code. **[Marks: 6]**

```
int main(){
        pid_t a,b;
        a=fork();
        if (a<0){
                printf("Fork failed\n");
        }
        else if (a==0){
                printf("Child started\n");
        }
        else{
                printf("Parent started\n");
                wait(NULL);
                printf("Child finished\n");
                b=fork();
                if(b<0){
                        printf("Fork failed\n");
                }
                else if (b>0){
                        printf("Parent started\n");
                        wait(NULL);
                        printf("Grand Child finished\n");
                        printf("Parent finished\n");
                }
                else{
                        printf("Grand Child started\n");
                }
                printf("Parent finished\n");
        }
        printf("terminating\n");
        return 0;
}
```



Outputs:

parent started
child started
terminating
child finished
parent started
Grand Child started
parent finished
terminating
Grand Child finished
Parent finished

parent finished
terminating