CS342 Operating Systems - Spring 2022
Homework #2

Assigned: March 2, 2022.
Due date: March 10, 2022, 23:59.

1) How many processes are created by the following pseudo-code:

```
main() {
        fork()
        fork()
        if (fork() == 0)
                fork()
        fork()
}
```

2) Assume speedup with 8 cores is 4. What is the speedup with 16 cores?

3) Consider the following code. What is printed out?

```
int x = 10;
main() {
        if (fork() == 0) {
                x = 20;
                if (fork()==0) {
                        x = 30;
                }
                print(x)
                x = 50;
        }
        print(x)
}
```

4) Schedule the following periodic processes using EDF (earliest deadline first). Show the 300 ms timeline.

P1: period 80, cputime=40.
P2: period 120; cputime=50.

5) We have the following processes. Schedule them with the following algorithms and find out the finish time, turnaround time , and waiting time, of each process. You need to consider priorities only in part e).

a) FCFS,
b) SJF,
c) SRTF (shortest remaining time first),
d) RR(q=20),
e) Preemptive priority scheduling with RR(q=20) applied on equal priority.

	Arrival time	CPU burst length (ms)	Priority
A	0	100	3
B	15	80	1
C	30	60	2
D	35	60	1
E	45	50	2

6) Assume exponential averaging is used in SJF to predict the next burst length. The α is 0.8. What is the predicted value for a process A after the following bursts were executed: 20ms, 10ms, 40ms, 30ms, 20ms. What is the predicted value for a process B after the following bursts were executed: 30ms, 20ms, 40ms, 10ms. Which one will be picked next to use the CPU? Why? You can use a calculator.

7) Consider the following program. Which values are printed out? Assuming an architecture where the size of an integer (`int`) is equal to the size of a pointer (`void *`).

```
int x = 100;
int y = 200;
int z = 300;
void *foo(void *arg)
{   int x;
    İnt k;
    x = x + 1000;
    y = y + 2000;
    printf (“%d %d %d\n”, x, y, (int)arg );
    return NULL;
}

int main()
{
    pthread_t tid;
    pthread_create(&tid, NULL, foo, (void *)z);
    pthread_join(tid, NULL);
    printf (“%d %d %d\n”, x, y, z);
    exit(0);
}
```