HW2 – Theoretical Questions:

1. Write two advantages for using a multiple-threaded process for performing a task over using a few processes.

2.

Consider a system running 3 I/O-bound processes, P1, P2 and P3. Each of them needs the CPU for 1 ms to make a request to an I/O device, which is served within 6 ms. This is done iteratively – namely, after the I/O request is fully served, each of them need again the CPU for 1 ms and so on.

Each of these processes calls a unique I/O device – namely, the I/O requests of T1, T2 and T3 may be served simultaneously.

The system also runs a single CPU-bound process, T4, which requires 100 ms of CPU, and doesn't use I/O devices.

A context switch (*ctxw)* takes 1 ms, in which the CPU performs only the ctxw, and can process nothing else.

We define as "CPU utilization" the percentage of cycles in which the CPU is processing a process - ie, the ratio between the number which the CPU is processing (i.e. not in idle / context switch mode), and the total number of cycles.

We assume that all processes arrive together, at time 0.

A) The scheduling policy is Preemptive Shortest Job First, and the quantum is 1 ms. Namely, after 1 ms, the scheduler schedules either another process, or the currently-running process, to run for the next 1 ms. What is the CPU utilization? What is the turnaround of T4?

B) In order to increase the CPU utilization, the quantum was increased to *n* ms, where *n* is a natural number. A currently-running process runs for *n* ms, unless it is blocked due to an I/O call.
What is the minimal *n* required for achieving a utilization of 70%? What is the turnaround of T4 in that case?

C) C) Repeat B when the desired utilization is at least 90%.

3.

In a system with a single processor and a NON-PREEMTIVE scheduler, there are four independent processes.

- Each process has a weight w_i that represents its priority order.
- In addition, each process has a runtime t_i and the processor prefers to run processes in descending order of weight.
- If two processes have the same weight, the process that arrived earlier (according to the Arrival Time) will run first.

$$w_4 = 4\ w_3 = 1\ w_2 = 3\ w_1 = 3$$
$$t_4 = 4\ t_3 = 8\ t_2 = 2\ t_1 = 5$$

- Arrival times:

$$a_1 = 0\ a_2 = 1\ a_3 = 2\ a_4 = 0$$

A. What is the order of execution of the processes considering the weights, arrival times and runtimes?

B. What is the turnaround time of each process?

C. How many different options for execution schedules are possible if the weights are not unique?

4.

In this question we consider the code in file ***HW2_24.c***.

Assume that no signal is lost – that is, every single is caught and handled by **every** process which may catch it.

It's given that during the time a process waits for 1[sec], the system finishes printing to the screen the outputs of all the other processes.

Answer the following American questions, and **explain your answer**. There can be more than one correct option for every question.

1. Upon running the program (without typing anything), which of the following outputs is possible?

A) Nothing is printed.

B)^CProcess number 3 caught one
Process number 2 caught one
Process number 1 caught one
Process number 0 caught one

C)^CProcess number 0 caught one
Process number 1 caught one
Process number 2 caught one
Process number 3 caught one

^CProcess number 0 caught one
Process number 1 caught one
Process number 2 caught one

^CProcess number 2 caught one
Process number 1 caught one
Process number 0 caught one

D) All the previous answers are wrong.

2. Yossi runs the program, and then presses ctrl+C. Which of the following outputs is possible?

A)^CProcess number 3 caught one Process number 2 caught one Process number 1 caught one Process number 0 caught one Process number 2 caught one Process number 1 caught one Process number 0 caught one Process number 1 caught one Process number 0 caught one Process number 0 caught one	B)^CProcess number 0 caught one Process number 1 caught one Process number 2 caught one Process number 3 caught one Process number 0 caught one Process number 1 caught one Process number 2 caught one Process number 0 caught one Process number 1 caught one Process number 0 caught one
C) ^CProcess number 3 caught one Process number 1 caught one	D)^CProcess number 0 caught one Process number 1 caught one

Process number 2 caught one Process number 0 caught one Process number 2 caught one Process number 0 caught one Process number 1 caught one Process number 1 caught one Process number 0 caught one Process number 0 caught one	Process number 3 caught one Process number 0 caught one Process number 1 caught one Process number 2 caught one Process number 2 caught one Process number 1 caught one Process number 0 caught one Process number 0 caught one
E) ^CProcess number 2 caught one Process number 1 caught one Process number 3 caught one Process number 0 caught one Process number 1 caught one Process number 2 caught one Process number 2 caught one Process number 1 caught one Process number 0 caught one Process number 2 caught one	F) ^CProcess number 0 caught one Process number 1 caught one Process number 2 caught one Process number 0 caught one Process number 1 caught one Process number 2 caught one Process number 3 caught one Process number 1 caught one Process number 0 caught one Process number 2 caught one

5.

At time 0 arrive processes 1, 2, 3, 4, requiring processing times 5,4,3,2 respectively.
Assume that process 1 arrived first, process 2 arrived 2nd, and so on.
Calculate the mean turnaround when using:

A) FIFO
B) RR with quanta=1
C) SJF

6.

Part I

At time 0 arrive processes 1, 2, 3, 4, requiring processing times 2,3,4,5 respectively.
Assume that process 1 arrived first, process 2 arrived 2nd, and so on.
Calculate the mean turnaround when using:

A) FIFO
B) RR with quanta=1
C) SJF

Part II

At time 0 arrive processes 1, 2, 3, 4,5 ,6 which require the following processing time: 3,2,5,10,2,1.
(the leftmost number is the process which arrives first).
For each of the following algorithms, calculate the average turnaround.

A. FIFO
B. RR with quanta = 1
C. RR with quanta = 2
D. SJF

7.

Consider a system with a preemptive RR scheduling with quanta = 1. The RR scheduler uses a cyclic pointer, which points to the next process to be executed. The pointer is initiated to 1, and is incremented by 1. If the pointed process doesn't need the CPU, the pointer is incremented again, until it reaches a process, which needs the CPU.
All processes arrive at time 0. Neglect the context switch time.
The processes use CPU and I/O alternately as follows:
Process 1: 1,3,1,3,3,1
Process 2: 3,2,3,1,3,2
Process 3: 1,4,1,3,2
For instance, 1 requires 1 CPU cycle, then 3 I/O cycles, then again 1 CPU cycle and so on. The system has 3 I/O devices, which may work on parallel (each serves a single request).

A. Plot a Gantt table.
   I. What is the utilization of the CPU and of the I/O devices until all the processes finish? For the utilization of the I/Os you have to divide the total number of I/O cycles by (3 times the number of cycles until all processes are done)
   II. What is the average turnaround time?
B. The system has a single I/O device, which uses priority-based preemptive scheduling, where the priority is defined by the process ID (process 1 before process 2, process 2 before process 3). Repeat A.

8.
Consider an OS with n processes, where process p_i is generated in time $t = i$. For instance, process 0 is generated in time $t = 0$, process 1 in time $t = 1$ … and process p_{n-1} is generated in time $t = n - 1$.
Each process requires exactly $3 \cdot n$ CPU cycles, and outputs for the first time after 3 cycles. Neglect the time required for output.

A. Suggest a scheduling policy, which minimizes the average response time. Justify your answer.
B. Is there any scheduling policy which minimizes both the average response time and the average turnaround time? Justify your answer.