Simulation Techniques

1. Task

The process scheduling (also called as **CPU scheduler**) is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. A scheduling allows one process to use the CPU while another is waiting for I/O, thereby making the system more efficient, fast and fair. In a multitasking computer system, processes may occupy a variety of states (Figure 1). When a **new** process is created it is automaticaly admitted the **ready** state, waiting for the execution on a CPU. Processes that are ready for the CPU are kept in a **ready queue**. A process moves into the **running** state when it is chosen for execution. The process's instructions are executed by one of the CPUs of the system. A process transitions to a **waiting** state when a call to an I/O device occurs. The processes which are blocked due to unavailability of an I/O device are kept in a **device queue**. When a required I/O device becomes idle one of the processes from its **device queue** is selected and assigned to it. After completion of I/O, a process switches from the **waiting** state to the **ready** state and is moved to the **ready queue**. A process may be **terminated** only from the **running** state after completing its execution. Terminated processes are removed from the OS.

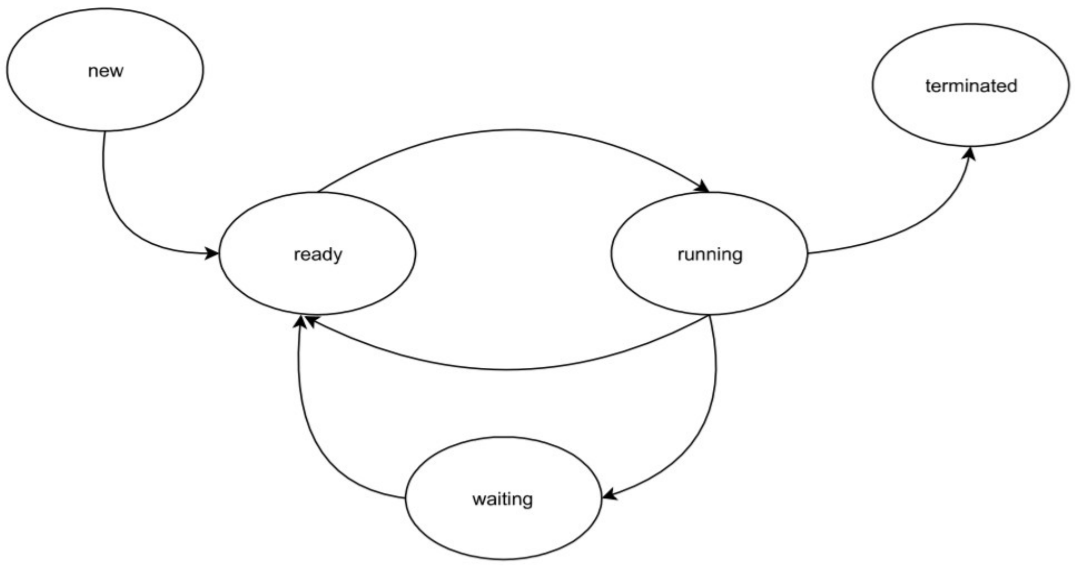

Figure 1. Process state diagram

Develop a C++ project that simulates the process scheduling described above, in accordance with the following parameters:

- Process Generation Time (PGT) [ms] – time before generation of a new processes (random variable with exponential distribution and intensity **L**) (round to natural number)
- CPU Execution Time (CET) [ms] – process execution time in CPU. Random variable with uniform distribution between <1, 50> [ms] (natural number)
- I/O Call Time (IOT) [ms] – time between getting an access to the CPU and an I/O call. Random variable with uniform distribution between <0, CET-1> [ms] (natural number). In case of 0, there is no I/O call.
- I/O Device (IOD) – indicates which I/O device is requested by the running process. Random variable with uniform distribution between <0, N_{IO}-1>, where N_{IO} is the number of I/O devices in the OS.

- I/O Time (IOT) [ms] – I/O occupation time. Random variable with uniform distribution between <1, 10> [ms] (natural number).

Determine the value of the parameter **L** that ensures the average waiting time in the ready queue not higher than 50 ms. Then, run at least ten simulations and determine:

- CPU utilization (for every CPU) [%]
- Throughput – number of processes completed (terminated) per unit time
- Turnaround time – time required for a particular process to complete, from its generation until termination [ms]

Draw a figure of an average waiting time in the ready queue in the function of parameter L.

	Simulation method
M1	Activity scanning
M2	Event scheduling
M3	ABC approach
M4	Process Interaction

CPU scheduling algorithms	**Description**
FCFS	In the "First come first serve" scheduling algorithm, as the name suggests, the process which arrives first, gets executed first, or we can say that the process which requests the CPU first, gets the CPU allocated first.
Random	In the random scheduling algorithm, the processes get executed in the random order, regardless of the arrival time.
SJF	Shortest job first is a scheduling policy that selects for execution process with the smallest execution time. Whenever a scheduling event occurs (a task finishes, I/O call occurs), the ready queue should be searched for the process with the smallest CPU execution time, which will be the next to be scheduled for execution. If two or more processes have the same CPU execution time, one of them should be chosen randomly (with the same probability each).

I/O scheduling algorithms	**Description**
FCFS	In the "First come first serve" scheduling algorithm, as the name suggests, the process which arrives first to the device queue, gets access first, or we can say that the process which requests the I/O first, gets the I/O allocated first.
Random	In the random scheduling algorithm, the processes get access in the random order, regardless of the arrival time.

SJF	Shortest job first is a scheduling policy that selects process with the smallest I/O occupation time. Whenever a scheduling event occurs (I/O becomes idle), the device queue should be searched for the process with the smallest I/O occupation time, which will be the next to get acess. If two or more processes have the same I/O occupation time, one of them should be chosen randomly (with the same probability each).