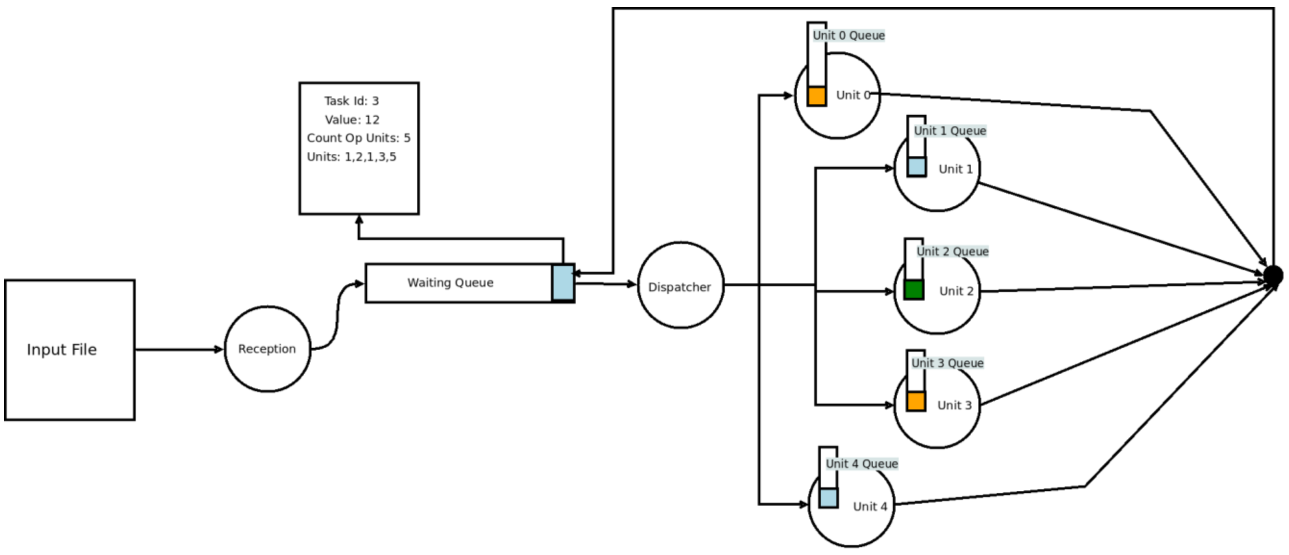
Namal University
Department of Computer Science
Operating Systems

Project

Total Marks: 50 *Topics: Interrupt, Process, Thread*

Q 01. Imagine a complex system that has several processing units for doing tasks. In this system, there is a receptor whose job is queueing input tasks by order of reading them from the input. Then, a dispatcher should send these tasks to the corresponding unit of the execution queue. Each unit of execution has its scheduling algorithm and does its operation on the task, and if there are other associated units of execution, the task is sent to the job queue, otherwise, the task is done. The following figure illustrates the system structure.



Task Structure

The structure of the task should have the following information. You are allowed to add other attributes to it if you find it necessary.

```
struct task {
    int id;            // id of this task
    int value;         // value of this task
    int atime;         // the arrival time of the task
    int unit_count;    // number of units
    int *unit_id;      // list of units which task will be assigned to
}
```

Units and Operations

Each unit of execution has its operation which will be done on the value of tasks. For example, unit 0 adds 7 to the value of the task and then modulates the new number with M. M is a constant value that should be defined in your program with a value of 10000. The reason for using modular division is to avoid value overflow. Each of these units are implemented as threads. Also, these units have their queue from which the tasks are picked according to their scheduling algorithm. The units should go to sleep for 0.5 seconds after doing the task.

Unit id	**Operation**	**Description**
0	**+7, %M**	**Adds 7 to the value then modulate by M**
1	***2, %M**	**Multiplies by 2 then modulate by M**
2	**^5, %M**	**Calculates the power 5 and then modulates by M**
3	**-19**	**Subtracts the value from 19**
4	**print**	**Prints out the value of the task before and after applying the operation**

Input file and Receptor

The Receptor should read an input file and construct struct of each task, and determine the arrival of the task, then put that task in the waiting queue. A thread-safe structure is required for implementing waiting queue. The structure of input file is as follows:

task-id task-value units-count unit-id-1 unit-id-2 ...

Example of Input file

0 123 5 4 0 0 4 2

1 78 3 1 2 4

2 -3 7 0 1 3 0 2 3 4

...

Print Unit

The output is generated by the print unit. The total time of being in the system until the print unit should be shown.

The format of out put should look like the following.

<time stamp> tid: <task_id> value: <value> value after operations: <value>

13 tid: 0 value: 123

18 tid: 0 value: 137

...

Note: This is the general structure of the system and you are free to make your own design choices where needed.

Submission Guidelines:

1. This is a group project and a maximum of 3 students are allowed in a group.
2. You must submit the .cpp file over QOBE and your code should be properly commented to explain the logic of the code.
3. **Submission Deadline: 21st January, 2024 11:59 PM**
4. You have to give a demo of your project in the form of a presentation.