Lab 5. Heaps

Release: **26th February 2024**
Supervision: **5th March 2024**
Due: **11th March 2024**

Vladimir Tarasov

1 Task: Printer Simulation 2

1.1 Introduction

Access to shared resources in computer systems is often controlled by a queueing mechanism. A common example is printers. In this lab task you will simulate the scenario of a laboratory printer (as shown in Fig 1).

1.2 Laboratory Printer Scenario

On average there will be one print task in N seconds. The length of the print tasks ranges from 1 to M pages. The printer in the lab can process P pages per minute, i.e., on average a printing task is finished in $pages * (60/P)$ seconds, at good quality. The scenaria is similar to the one presented in the lab 2, but applies a priority queueing mechanism. The printer always processes the printing task with the *smallest number of pages* first. If two tasks have the same number of pages, the printer processes the tasks on a first come first served (FCFS) basis.

The simulation runs roughly as the steps as in the simulation presented in the lab 2.

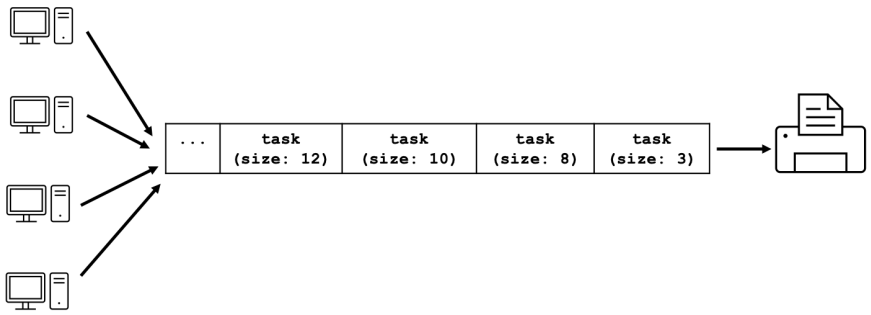

Figure 1: Laboratory printer scenario

Simulation of a laboratory printer

1. Set the printer state to empty.
2. Create a queue for handling print tasks.
3. While (the clock limit is not reached)
   - Is a new print task created? If so, add it to the queue with the current second as the timestamp upon its arrival.
   - If the printer is not busy and if a task is waiting, remove the front task from the queue and assign it to the printer.
   - The printer now does one second of printing if necessary, i.e., subtracts one second from the time required for the task. If the task has been completed, in other words the time required has reached zero, the printer is no longer busy.
4. Simulation ends.

1.3 Requirements

You should implement the interfaces for printer ADT, task ADT and priority queue ADT as defined in the header files (in printer_simulation_headers_priority_queue.zip), and a main function for the simulation process. As defined in the priority queue ADT interface, the queue should be implemented using a *min binary heap*. The binary heap should be represented as an *array*.

Hints

1. You have learned how to implement a max binary heap using an array in the exercise. Revise the program to build a min binary heap to handle printing tasks.
2. Reuse the printer, task and the main functions you implemented in lab 2.
3. Check whether the priority queue of printing tasks is correctly represented as an array for implementation of the min binary heap.

Once you have got your program implemented correctly, you should be able to produce output like the example shown below, where $N = 1$, $M = 10$, $P = 80$, the simulation runs for 10 seconds and the array size is 20. You can choose different values for the simulation parameters in your program.

Example program output

```
Simulation starts ...

simulation at second 1
 ------------------------
THE PRINTER IS NO LONGER BUSY
TASKS IN QUEUE: [task: 7 pages, arrives at 1s],
take the front task from the priority queue
NO TASK IN QUEUE
printing the current task ...

simulation at second 2
 ------------------------
THE PRINTER IS BUSY
4 seconds to complete the current task.
TASKS IN QUEUE: [task: 6 pages, arrives at 2s],
printing the current task ...

simulation at second 3
 ------------------------
THE PRINTER IS BUSY
3 seconds to complete the current task.
TASKS IN QUEUE: [task: 6 pages, arrives at 2s], [task: 6 pages, arrives at 3s],
printing the current task ...

simulation at second 4
 ------------------------
THE PRINTER IS BUSY
2 seconds to complete the current task.
TASKS IN QUEUE: [task: 3 pages, arrives at 4s], [task: 6 pages, arrives at 3s], [task: 6 pages, arrives at 2s],
printing the current task ...

simulation at second 5
 ------------------------
THE PRINTER IS BUSY
1 seconds to complete the current task.
TASKS IN QUEUE: [task: 2 pages, arrives at 5s], [task: 3 pages, arrives at 4s], [task: 6 pages,
 arrives at 2s], [task: 6 pages, arrives at 3s],
printing the current task ...

simulation at second 6
 ------------------------
THE PRINTER IS NO LONGER BUSY
TASKS IN QUEUE: [task: 2 pages, arrives at 5s], [task: 3 pages, arrives at 4s], [task: 6 pages,
 arrives at 2s], [task: 6 pages, arrives at 3s], [task: 8 pages, arrives at 6s],
take the front task from the priority queue
TASKS IN QUEUE: [task: 3 pages, arrives at 4s], [task: 6 pages, arrives at 3s], [task: 6 pages, arrives at 2s], [task: 8 pages,
 arrives at 6s],
printing the current task ...

simulation at second 7
 ------------------------
THE PRINTER IS NO LONGER BUSY
TASKS IN QUEUE: [task: 3 pages, arrives at 4s], [task: 6 pages, arrives at 3s], [task: 6 pages, arrives at 2s], [task: 8 pages,
```

```
 arrives at 6s], [task: 10 pages, arrives at 7s],
take the front task from the priority queue
TASKS IN QUEUE: [task: 6 pages, arrives at 2s], [task: 6 pages, arrives at 3s], [task: 10 pages, arrives at 7s], [task: 8 pages,
 arrives at 6s],
printing the current task ...

simulation at second 8
 ------------------------
THE PRINTER IS BUSY
1 seconds to complete the current task.
TASKS IN QUEUE: [task: 6 pages, arrives at 2s], [task: 6 pages, arrives at 3s], [task: 10 pages, arrives at 7s], [task: 8 pages,
 arrives at 6s], [task: 7 pages, arrives at 8s],
printing the current task ...

simulation at second 9
 ------------------------
THE PRINTER IS NO LONGER BUSY
TASKS IN QUEUE: [task: 6 pages, arrives at 2s], [task: 6 pages, arrives at 3s], [task: 7 pages, arrives at 9s], [task: 8 pages,
 arrives at 6s], [task: 7 pages, arrives at 8s], [task: 10 pages, arrives at 7s],
take the front task from the priority queue
TASKS IN QUEUE: [task: 6 pages, arrives at 3s], [task: 7 pages, arrives at 8s], [task: 7 pages, arrives at 9s], [task: 8 pages,
 arrives at 6s], [task: 10 pages, arrives at 7s],
printing the current task ...

simulation at second 10
 ------------------------
THE PRINTER IS BUSY
3 seconds to complete the current task.
TASKS IN QUEUE: [task: 6 pages, arrives at 3s], [task: 7 pages, arrives at 8s], [task: 7 pages, arrives at 9s], [task: 8 pages,
 arrives at 6s], [task: 10 pages, arrives at 7s], [task: 7 pages, arrives at 10s],
printing the current task ...

SIMULATION ENDS
```

1.4 Deliverables

The deliverable is a zip file in which the files are organized as in Fig 2. Variables in your code files should have proper names, and the code has to include sufficient comments for readability.

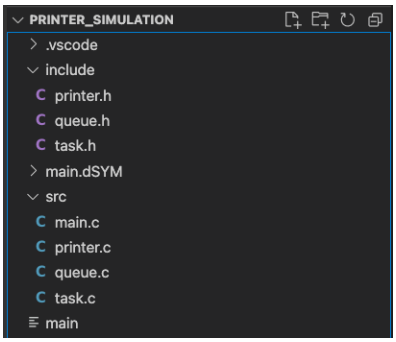

Figure 2: File organization

The provided header files *must be included* in your implementation. It is up to you to decide how you code the `main()` function.

The source code has to be thoroughly tested as well as to adhere to the recommendations on *Standard C and portability*, which you can find in Canvas.

You can compile your code using more strict flags `-Wall` together with `-pedantic` before submitting the code, e.g.

```
$ gcc -Wall -pedantic program.c -o program
```

Please include a comment in your program (all `.c` files) with the year and your name to indicate authorship:

```
// 2024 Your Name
```