10710EECS204001  
Data Structures Homework 1

Due date: 2018/10/16 23:59

Submit to OJ: #12014

Upload code to iLMS

Submission

* Please **1)** submit your code to OJ (OJ: #12014),   
  and **2)** upload the zipped file (source codes) to iLMs.   
  **Both should be done before the due date.**
* Scores will be given based on your OJ results, and the uploaded zipped file (the source codes) should be identical to those submitted to OJ. TAs will examine your uploaded codes.

Description

OS CPU Scheduling: Round-robin

Round-Robin (RR) is a common scheduling algorithm for CPU scheduling. Each process uses the CPU for a period of time. The length of this time is called *time quantum*. We setup a “ready queue” for the process waiting to use the CPU. When the CPU is not used, we choose the first process in the ready queue to use the CPU.

Each process has the following information: **process\_id**, **arrival\_time**, and **require\_time**. **Arrival\_time** is the time that the process enters into the system, and **process\_id** is from small to large according to the **arrival\_time**. **Require\_time** is the time that the process requires to use the CPU.

If the process runs out of the time quantum and does not finish, the process will return to the ready queue and wait for the next time to use the CPU. In contrast, if the process completes before the time quantum runs out, we select the first process in the ready queue to use the CPU.

Please note:

1. If two or more processes enter the ready queue at the same time, the process with the smaller **process\_id** has enter the ready queue earlier.

2. In this homework, there is only one CPU in the system.

The following is a simple example of the Round-Robin algorithm, assume that the time quantum is 3.

|  |  |  |
| --- | --- | --- |
| process\_id | arrival\_time | require\_time |
| 0 | 0 | 5 |
| 1 | 2 | 2 |

At time 0, process 0 arrives and no other process is using the CPU. Therefore, process 0 can use the CPU directly.

At time 2, process 1 arrives. Since process 0 is using the CPU and does not run out of the time quantum (which is 3), process 1 enters the ready queue and waits.

At the time 3, process 0 runs out of the time quantum, so it returns to the ready queue. Since the first process in the ready queue is process 1, now it’s process 1’s turn to use the CPU.

At time 5, process 1 finishes using the CPU, and the first process in the ready queue is process 0. So it is process 0’s turn to use the CPU.

At time 7, process 0 finishes using the CPU.

In this example, the ending sequence of the processes is process 1→ process 0.

**Task:** Given n processes with their **process\_id**s, **arrival\_time**s and **require\_time**s, please implement the RR scheduling with time quantum = 3, and output the finishing order of the processes.

Note that using <stack>, <queue>, <vector> and <list> in C++ are not allowed.

**Time Limit:** 3 sec

Input

The input starts with an integer **n**, and is followed by **n** lines. Each line has three integers, **process\_id**, **arrival\_time,** and **require\_time**, which are separated by a single space.

(n<100, arrival\_time<999, require\_time<300)

Output

**Process\_id**s ordered by the finishing time, separated by commas.

You need to output a '\n' in the end of line

Sample Input

3

0 0 5

1 0 2

2 1 3

2

0 0 4

1 4 3

Sample output

1,2,0

0,1