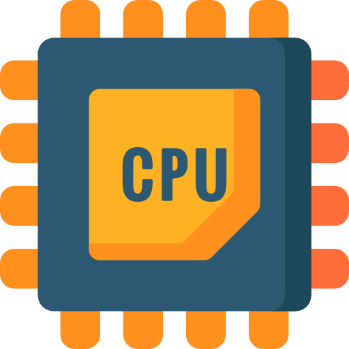
# Scheduling



*You are hired to create a program that implements SJF (Shortest Job First). It works by letting the shortest jobs to take the CPU, so jobs won't get frozen.*

On the **first line** you will be given the **jobs** as **integers** (clock-cycles needed to finish the job) separated by **comma and space** **", "**. On the **second line** you will be given the **index** of the job that we are interested in and want to know **how many cycles** will pass until the job is done.

The tasks that need the **least amount of clock-cycles** will be completed **first**.

For the jobs that need the **same amount** of clock-cycles, the order is **FIFO** (First In First Out).

You have to **print** how many **clock-cycles** will pass until the task you are interested in is **completed**. For more clarifications, see the examples below.

### Input

* On the first line you will receive **numbers** separated by **", "**
* On the second line you will receive the **index** of the task you are interested in

### Output

* Single line: the **clock-cycles** that will **pass** until the task you are interested in is **finished**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 3, 1, 10, 1, 2  0 | 7 | The first task will be 1 at index 1 (1 clock-cycle)  Next is 1 at index 3 (total 2 clock-cycles)  Next is 2 at index 4 (total 4 clock-cycles)  Next, we arrive at 3 on index 0 (total 7 clock-cycles) which is the one we need, and we end the program |
| 4, 10, 10, 6, 2, 99  2 | 32 | 2 at index 4 -> total 2 clock-cycles  4 at index 0 -> total 6 clock-cycles  6 at index 3 -> total 12 clock-cycles  10 at index 1 -> total 22 clock-cycles  10 at index 2 -> total 32 clock-cycles |

*I burned my finger on my computer processor…it MHz!*