## Introduction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | problem **0** | | **Queueing  Theory** | | y points | |  |
|  |  |

Queueing theory is the mathematical study of waiting in lines. Doesn't that sound fun? In computing, queues are used to track requests that have been issued but not started. This is seen in applications like web services or online games in which a large number of people request data from a server. The order of request processing we will consider is called First-In First-Out or FIFO. With a FIFO the processing works like lines at a grocery store, where each line (queue) is filled with customers (inbound requests) and is serviced by a cashier (a time-constrained processor). Customers arrive at the back of the line and the cashier services the customer at the front of the line. In other words, FIFO.

Simulating a time-constrained process in a contest setting is awkward. Instead, we'll use a process that provides a simple way to verify that the FIFOs are working correctly. Your program will simulate system storage using a data string that has been initialized with spaces. Queued requests will consist of a word and an offset. The program must write the requested word into the data string at the designated offset. The first character in the data string is at offset zero. All requests write into the same data string buffer. As an example of a request, for a data string of length 44 the request “35 KNOWN” would result in the following data string:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | K | N | O | W | N |  |  |  |  |

Each request is added to a request queue and processed later. Some requests may overwrite portions of words previously written in the data string, so sequence is important. Each queue is a FIFO, but just like at the grocery store multiple queues can mean that requests (customers) are not always processed in the order they arrived.

# Sample Input

The first line contains the length L of the data string and the number N of requests. The next N lines are the actual requests. Each request has three parts: a queue number, an offset, and a word. There may be up to nine request queues (amazingly, numbered 1 through 9). The last line of input designates the order in which the requests are actually processed from the queues.

44 13  
Q1 35 KNOWN  
Q1 20 IMPORT  
Q3 24 GRANT  
Q1 4 IN  
Q1 15 MADE  
Q1 32 AN  
Q2 39 LEDGE  
Q2 5 NOTION  
Q2 6 A  
Q2 16 OR  
Q3 0 IMAGE  
Q3 12 IS  
Q3 30 THIS  
Q1 Q3 Q3 Q3 Q2 Q1 Q2 Q1 Q3 Q1 Q2 Q2 Q1

# Sample Output

The program must process each request in the designated order and print the result string.

IMAGINATION IS MORE IMPORTANT THAN KNOWLEDGE