D. Parking Lot

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Nowadays it is becoming increasingly difficult to park a car in cities successfully. Let's imagine a segment of a street as long as L meters along which a parking lot is located. Drivers should park their cars strictly parallel to the pavement on the right side of the street (remember that in the country the authors of the tasks come from the driving is right side!). Every driver when parking wants to leave for themselves some extra space to move their car freely, that's why a driver is looking for a place where the distance between his car and the one behind his will be no less than b meters and the distance between his car and the one in front of his will be no less than f meters (if there's no car behind then the car can be parked at the parking lot segment edge; the same is true for the case when there're no cars parked in front of the car). Let's introduce an axis of coordinates along the pavement. Let the parking lot begin at point 0 and end at point L. The drivers drive in the direction of the coordinates' increasing and look for the earliest place (with the smallest possible coordinate) where they can park the car. In case there's no such place, the driver drives on searching for his perfect peaceful haven. Sometimes some cars leave the street and free some space for parking. Considering that there never are two moving cars on a street at a time write a program that can use the data on the drivers, entering the street hoping to park there and the drivers leaving it, to model the process and determine a parking lot space for each car.

Input

The first line contains three integers L, b и f ($10 \le L \le 100000$, $1 \le b, f \le 100$). The second line contains an integer n ($1 \le n \le 100$) that indicates the number of requests the program has got. Every request is described on a single line and is given by two numbers. The first number represents the request type. If the request type is equal to 1, then in that case the second number indicates the length of a car (in meters) that enters the street looking for a place to park. And if the request type is equal to 2, then the second number identifies the number of such a request (starting with 1) that the car whose arrival to the parking lot was described by a request with this number, leaves the parking lot. It is guaranteed that that car was parked at the moment the request of the 2 type was made. The lengths of cars are integers from 1 to 1000.

Output

For every request of the 1 type print number -1 on the single line if the corresponding car couldn't find place to park along the street. Otherwise, print a single number equal to the distance between the back of the car in its parked position and the beginning of the parking lot zone.

Examples

input
30 1 2 6 1 5 1 4 1 5 2 2 1 5 1 4

output
0 6 11 17 23

input
30 1 1 6

```
1 5
1 4
1 5
2 2
1 5
1 4
```

output

```
0
6
11
17
6
```

input

```
10 1 1
1
1 12
```

output

```
-1
```