"Efficient" Crowd Control

locked

At a railway station, there are 3 ticket counters, identified as $C1$, $C2$ and $C3$ respectively. Also, there are N persons who need to buy tickets on a particular day, where each person is identified by an unique ID number from 1 to N.

As part of the input, you are provided with a sequence A, having N elements, where the i-th element A_i denotes the number of tickets that the person with ID number i $(1 \leq i \leq N)$ needs to buy.

As part of the input, you are also provided with a sorted sequence T, having N elements, where the i-th element T_i denotes the time (counted in seconds elapsed from the start of the day, i.e. from 00:00:00hrs) at which the person with ID number i $(1 \leq i \leq N)$ arrives at the railway station.

The rules that each person needs to follow for buying tickets at the railway station are:

- Upon their arrival at the railway station, he/she must initially join the queue at the $C1$ counter.
- If two different persons arrive at the railway station at the same time, i.e. if we have $T_i = T_j$ for some pair (i, j) where $i \leq j$, then the person with ID number i will get to join the queue at the $C1$ counter before the person with ID number j.
- On reaching the $C1$ counter (along the queue), he/she is only allowed to buy at most k_1 tickets.
- After having just bought tickets from the $C1$ counter, if he/she still needs to buy more tickets, then he/she must join the queue at the $C2$ counter.
- On reaching the $C2$ counter (along the queue), he/she is only allowed to buy at most k_2 tickets.
- After having just bought tickets from the $C2$ counter, if he/she still needs to buy more tickets, then he/she must join the queue at the $C3$ counter.
- On reaching the $C3$ counter (along the queue), he/she is only allowed to buy at most k_3 tickets.
- After having just bought tickets from the $C3$ counter, if he/she still needs to buy more tickets, then he/she must join the queue at the $C1$ counter.
- If there are two or more persons who are ready to join the queue for counter C_1 at the same time, then the newly arrived person(s), i.e. the person(s) having the largest T_i, gets to join the queue before the person who just finised buying tickets from the C_3 counter.

Every person will keep following the above set of rules, until he/she has bought the desired number of tickets, at which point they will leave the station.

The server at each counter takes exactly one second to issue tickets to the person who has just reached the counter by moving along the queue. The servers at all the counters work in parallel, and are independent of each other.

Finally, you are given a sorted sequence Q, containing q natural numbers, where the i-th number Q_i denotes a query time (counted in seconds elapsed from the start of the day, i.e. from 00:00:00hrs). For every $i \in \{1, 2, \ldots, q\}$, your task is to print the ID numbers of all the persons (in order) standing in the queues in front of each of the three counters at time Q_i.

Input Format

The first line contains a single integer - N

The second line contains N space-separated integers - $A_1, A_2, \ldots, A_N$

The third line contains N space-separated integers - $T_1, T_2, \ldots, T_N$

The fourth line contains 3 space-separated integers - k_1, k_2, k_3

The fifth line contains a single integer - q

The sixth line contains q space-separated integers - $Q_1, Q_2, \ldots, Q_q$

Constraints

$1 \leq N \leq 200$

$1 \leq A_i \leq 1000$

$0 \leq T_i \leq 1000$

$1 \leq k_1, k_2, k_3 \leq 1000$

$1 \leq q \leq 100$

$0 \leq Q_i \leq 1000$

Output Format

For each query time Q_i, you need to print 3 lines of output, where:

First line contains the status of first counter queue - indexes of people standing in order

Second line contains the status of second counter queue - indexes of people standing in order

Third line contains the status of third counter queue - indexes of people standing in order

Note:- For each counter C_i, the first number on each line denotes the ID number of the person at the front of the corresponding queue, i.e. the person closest to the counter C_i

Note:- For each counter C_i, you should print just "-1" on the line if the corresponding counter queue is empty.

Sample Input 0

```
3
4 4 4
```

```
0 1 5
2 2 2
2
0 1
```

Sample Output 0

```
-1
1
-1
-1
2
-1
```

Explanation 0

At time t=0, Person 1 joins at the first counter (c1).

They then buy 2 tickets, and move on to counter 2 (c2).

Since the other counters were empty, no other changes are made.

These are all the operations done during time t=0 (before t=1 starts)

Hence, at the end of time = 0:

c1 = empty

c2 = (start) 1 (end)

c3 = empty

At time t=1, Person 2 joins at the first counter (c1).

They then buy 2 tickets, and move on to counter 2 (c2).

Now, Person 1 waiting at counter 2 also gets to buy 2 tickets. Since they only wanted 4 tickets, they leave and do not wait at counter 3 (c3).

These are all the operations done during time t=1 (before t=2 starts)

Hence, at the end of time = 1:

c1 = empty

c2 = (start) 2 (end)

c3 = empty

Submissions: 85
Max Score: 100
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

Current Buffer (saved locally, editable)   C++

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}

```

Line: 1 Col: 1

Upload Code as File   Test against custom input   Run Code   Submit Code

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature