Operating System Lab 9

Task-1: Implement Banker's algorithm for deadlock avoidance.

Given three resources A,B,C and total availability of all the resources x,y,z respectively. Find a valid sequence for which system does not give deadlock. if no such sequence exists return -1. if one such sequence exists through banker's algorithm also return all possible sequences that does not result in deadlock.

Input Description

The first line of the input contains one integer t (1≤t≤10^4) — the number of test cases.

Each test case consists of following n+2 lines:

- First line contains one integer n (1≤n≤4·10^5) — the number of processes
- Second line contains three integers x,y,z (1<=x,y,z<=10^5) where x,y,z are the amount of total resources available for A,B,C.
- Next n lines contain x1, x2, x3, y1, y2, y3 (1<=xi, yi<=10^5) where (x1, x2, x3) are currently allocated resources to process pi and (y1, y2, y3) are max total requirement of the process pi for the resources A, B, C.

Sample Input:

1

5

3 3 2

0 1 0 7 5 3

2 0 0 3 2 2

3 0 2 9 0 2

2 1 1 2 2 2

0 0 2 4 3 3

Process	Allocation	MAX	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Output Description

for each test case print the sequence through banker's algorithm and then print all possible sequences valid for the test case (comma separated sequences, space separated numbers).

Sample Output

1 3 0 2 4,1 3 0 4 2,1 3 2 0 4,1 3 2 4 0,1 3 4 0 2,1 3 4 2 0,1 4 3 0 2,1 4 3 2 0,3 1 0 2 4,3 1 0 4 2,3 1 2 0 4,3 1 2 4 0,3 1 4 0 2,3 1 4 2 0,3 4 1 0 2,3 4 1 2 0

Task-2: Implement DS most recently used (MRU) page replacement algorithm.

Unlike LRU, MRU page replacement algorithm replaces the most recently used page. In this task, we design a data structure that works like a MRU page replacement. Here cap denotes the capacity of the pages in ram and Q denotes the number of queries. Query can be of two types:

SET x y: sets the value of the key x with value y

GET x: gets the key of x if present else returns -1.

Implement 2 functionalities:

get(key): returns the value of the key if it already exists in the pages otherwise returns -1.

set(key, value): if the key is already present, update its value. If not present, add the key-value pair to the pages. If the ram reaches its capacity it should invalidate the most recently used item before inserting the new item.

Input

The first line of the input contains one integer t ($1 \leq t \leq 10^4$) — the number of test cases.

Each test case consists of 3 lines:

- the first line contains one integer cap (1≤cap≤4·10^5) capacity of the pages in ram
- the second line contains one integer Q (1≤Q≤4·10^5) number of queries
- the third line contains Q queries in the given format q1,q2,q3...where qi can be ‘s’ xi,yi or ‘g’ xi (1<=xi,yi<=10^4)

Output

for each test case output is k integers where k=number of get queries in qi

example:

Input:

1

2

2

S 1 2 G 1

Output:

2