MMM '14 H - Lift Control

Time limit: 1.0s **Memory limit:** 256M

Maniacal Midsummer Marathon 2014 by AL, TL, JJ

Fun fact: Do you like mashing on those door-close buttons in elevators? Well, in most elevators built since the 90s, those buttons are just there to give passengers a false sense of control.

You were noticeably enraged when your friend, the lead engineer working for a large elevator firm, told you this fact. "What is this bamboozlement I've been experiencing my whole life?" you ask him. "Elevators are always so slow and lead to such long waiting times. And you blame *us*, the *passengers* for always being in a hurry and slamming on the door-close buttons. Can't *you* guys just design better elevators?"

He responds by telling you that behind-the-scenes, elevator companies work really hard to make elevators really efficient. The best elevator algorithm still remains a very difficult open problem. Companies even keep their own algorithms a trade secret.

That's when your entitled ass started boasting to him about easily being able to design a way better algorithm than whatever garbage his incompetent team can come up with. You quickly realize that this was a big mistake, because he actually challenged you to an elevator face-off. Being the lead engineer, he actually has the resources to pull this off. He claims that he'll give you a job at his firm if you manage to win or tie against him, but that you'll have to wax his hairy back if you lose. Being the sad, unemployed nerd you are, living with your parents and eating Cheetos off your chest every day, you have no choice but to accept his challenge to finally try and land yourself a job.

The challenge is to write a program for the computer of the elevator in a building, such that **the average waiting time for passengers is minimized**. There will be cameras on each floor of the building to record the waiting times of passengers. First, he'll hook your algorithm up to run for a period of time and record the results. Then, your friend will take the passenger data collected from your service period, enter it into his special elevator simulator software, and run his own company's algorithm on it. In the end, the average waiting time of your algorithm and his algorithm will be compared.

After digging through some articles online, you discovered that - big surprise! - elevator systems actually do require quite a bit of complicated mathematical theory. There's no way you'll be able to learn it all in time. Just as you were heading out to buy some wax paper, a dastardly thought suddenly goes through your mind – what if you rigged the challenge?

The day before your contest, you've secretly paid off the building manager, having him make an announcement to everyone that the elevators are out of service for the time period of your bet. You also told him to seal off the doors so that only certain people can enter and use the elevators. Why? Well, you've secretly arranged for N passengers to use the elevator during the challenge. You'll know the exact times and floors related to their elevator usage so that you can rig your algorithm accordingly. Since your engineer friend has no clue about this, he'll have to rely on his algorithm being efficient during the actual service, while your algorithm will secretly have the traffic data way beforehand. What a perfect trick!

Your program will have the following information before the challenge:

- There are F $(1 \le F \le 1\,000)$ floors in the building being serviced, uniquely labeled with an integer from 1 to F.
- The elevator may change directions at any time. The time it takes to change directions is negligible.

- The elevator travels at a constant speed of V ($0 < V \leq 20$, V is a real number) floors per second. While the elevator is traveling, it may stop at any floor, either to remain idle, or to pick up/drop off passengers.
- The elevator may stop for as long as possible on a floor. Standard regulations require elevator doors to open for at least S ($1 \leq S \leq 20$, S is an integer) seconds each time. This means that if an elevator is told to stop for fewer than S seconds, it will simply remain idle with its doors closed, and no passengers may get on or off. Note that your passengers are bribed, so they will take no more than S total seconds to get on and off. They will never delay the door-close by hitting the door-open button or standing in the way of the sensors.
- There are N $(1 \leq N \leq 1\,000)$ passengers that need to be serviced. The passengers are each uniquely labeled with an integer from 1 to N.
- The following information is known about the passengers you've bribed:
  - At t_i seconds ($0 \leq t_i \leq 1\,000\,000$, t_i is an integer), passenger i $(1 \leq i \leq N)$ will arrive at the elevator doors on floor A_i $(1 \leq A_i \leq F)$.
  - Passenger i would like to go to floor B_i $(1 \leq B_i \leq F, A_i \neq B_i)$.
  - Passengers will be patient. That is, a passenger will never abandon their waiting spot until the elevator arrives.
  - If the elevator doors are open on a passenger's initial floor A_i, then the passenger will immediately step into the elevator.
  - Similarly, if a passenger is on the elevator which is currently on their designated floor B_i with its doors open, then the passenger will **immediately** step out of the elevator. For example, if the elevator doors open at $t = 10$s, then we consider the time of all passengers getting off during this stop to also be 10s.
  - If an elevator door opens at t_o and closes at t_c, and a passenger arrives at t_i during that range, then they may enter the elevator if and only if $t_o \leq t_i < t_c$.

The *waiting time* experienced by passenger i is the time from when they arrive, t_i, to the time when they get off (i.e. the moment the elevator doors open on their designated floor B_i).
The *average waiting time* is the sum of the waiting times of all of the passengers, divided by the number of passengers N.

Initially, the elevator is on floor 1 (closed) and the time is 0. Your program must output commands to direct the elevator, and must eventually move all the N passengers to their designated floors. Furthermore, your program should try to service all passengers such that the average waiting time of all passengers is minimized. In the end, your average end time will be scored against your friend's. Only when you beat or tie him will you be safe from waxing his back. The odds are already in your favor, so you'd better not screw up!

Input Specification

The 1^{st} line will contain the numbers F, S, and V, respectively representing the number of floors in the building, the minimum time that the elevator must open, and the speed of the elevator in floors/second. F and S will be integers, while V will be a real number.
The 2^{nd} line will contain the integer N, representing the number of passengers.
The following N lines each contain three integers describing the request of a passenger. Namely, line $i + 2$ will contain the values t_i, A_i, and B_i, indicating that passenger i starts his or her wait at time t_i by the elevator doors on floor A_i, and would like to be taken to floor B_i.

Output Specification

In order to control the elevator, you must first become familiar with the two commands of the elevator's system. They are the following:

Command	Format	Description
GO *b*	`G b`	Move the elevator from its current floor to floor b ($1 \leq b \leq F$, b is an integer) at a speed of V floors per second. If the elevator arrives on floor b at a non-whole number of seconds, then it will wait until the entire second is over before processing more commands. Specifically, if the current floor is a and the current time is t_1, then the elevator will stop on floor b at time $t_2 = t_1 + \left\lceil \frac{\lvert b-a \rvert}{V} \right\rceil$, where $\lvert x \rvert$ is the absolute value function, and $\lceil x \rceil$ is the smallest integer greater than or equal to x.
STOP *t*	`S t`	Stop the elevator for t ($0 \leq t \leq 1\,000\,000$, t is an integer) seconds. If t is less than the minimum regulated time S, then the elevator will simply idle on its current floor and passengers will *not* be able to get on for this stopped period. Otherwise if $t \geq S$, then the elevator doors will open for t seconds, allowing passengers to get on or off at the floor it's currently on. If the current time is t_1, then passengers arriving may step into the elevator during any time in the range $[t_1, t_1 + t)$ — that is, including t_1 but not including $t_1 + t$ (when the door closes).

Each line of the output should describe an elevator command in one of the two formats listed above. The commands will be performed back to back (with the slight exception of those that follow a `GO` command, which will be performed at the next available whole number second after the elevator arrives at its floor), and will control the behavior of the elevator from start to finish.

Sample Input

```
10 2 3.0
4
0 2 5
2 1 10
4 5 10
21 10 4
```

Sample Output

```
S 3
G 2
S 2
G 5
S 2
G 10
S 11
G 4
S 2
```

Explanation

There are 10 floors in the building. The elevator must stay open for at least 2 seconds every time it opens. The elevator moves at a rate of 3 floors per second. There are 4 passengers, respectively arriving at times 0, 2, 4, and 21. The execution of the output commands are as outlined in the following table.

Time	Command	Description	Passenger Status			
			1	2	3	4
0	`STOP 3`	The elevator is on floor 1. It opens for 3 seconds to wait for passenger 2. Meanwhile, passenger 1 has arrived and now waits on floor 2.	A	▨	▨	▨
1				▨	▨	▨

2		Passenger 2 arrives on floor 1, immediately entering the elevator just before it closes.		🟧 E	▨	▨
3	`GO 2`	The elevator travels at 3 floors per second, so going up by 1 more floor will require $1/3$ of a second. Thus, the elevator will only take $\frac{1}{3}$ seconds rounded up, which is 1 second.		🟧	▨	▨
4	`STOP 2`	The elevator is on floor 2. It opens for 2 seconds, and the waiting passenger 1 enters.	🟧 E	🟧	🟨 A	▨
5			🟧	🟧	🟨	▨
6	`GO 5`	The elevator travels to floor 5 from floor 2 (3 floors total). Since its speed is 3 floors/second, its total time to reach floor 5 will be 1 second.	🟧	🟧	🟨	▨
7	`STOP 2`	The elevator is on floor 5. It opens for 2 seconds. During this second, passenger 1 gets off. Also, passenger 3 gets on.	🟧	🟧	🟧 E	▨
8			▨	🟧	🟧	▨
9	`GO 10`	The elevator goes to floor 10 from floor 5. This takes $\lceil \frac{10-5}{3} \rceil = 2$ seconds.	▨	🟧	🟧	▨
10			▨	🟧	🟧	▨
11	`STOP 11`	The elevator is on floor 10. It opens for 11 seconds to allow both passengers 2 and 3 to get off. Afterwards, the elevator idles to wait for passenger 4 to arrive at $t = 21$.	▨	🟧	🟧	▨
...			▨	▨	▨	▨
21		Passenger 4 arrives and enters immediately.	▨	▨	▨	🟧 E
22	`GO 4`	6 floors will take $\lceil \frac{10-4}{3} \rceil = 2$ seconds to traverse.	▨	▨	▨	🟧
23			▨	▨	▨	🟧
24	`STOP 2`	The elevator doors open on floor 4. Passenger 4 exits immediately.	▨	▨	▨	🟧
25			▨	▨	▨	▨

In the above table: A means that the passenger has arrived and is waiting at their initial floor, and E means that the passenger is inside the elevator.

Passenger 1 waits during the range $[0, 7]$ for a total of 8 seconds.
Passenger 2 waits during the range $[2, 11]$ for a total of 10 seconds.
Passenger 3 waits during the range $[4, 11]$ for a total of 8 seconds.

Passenger 4 waits during the range $[21, 24]$ for a total of 4 seconds.
The average waiting time is $\frac{8+10+8+4}{4} = 7.5$ seconds. Note that this may not be the best possible solution for this scenario.

Scoring

For each test case, let's say the average waiting time for passengers during your lift service is X seconds, and the average waiting time for your friend's algorithm is Y seconds.
Then, your score out of 100 for the test case will be:

- 0, if any of the following holds true:
  - The output format is incorrect (lines do not follow one of the two formats listed above); or
  - Not all passengers have an opportunity to get on and off the elevator to reach their destination.
- between 10 and 100, if $X \geq Y$.
  - Your program scores 10 base points for getting any valid way of moving all passengers to their designated floors, plus an additional score that is determined from a rational scale based on the values X and Y.
  - Specifically, your score will be $10 + 90 \times \frac{Y}{X}$, rounded to the nearest integer.
- 100 (or over), if $X < Y$.

Here, Y will actually be the average waiting time for the test case obtained by the current submission with the highest score. Thus, the cases will be updated periodically in accordance with the current best submission (perhaps until a very nice solution is found).

Experimentation

For testing purposes, you are provided with a program to simulate the activities of the elevator and passengers. The simulator program simulator.py will accept the following command line options:

- `-h` will give a brief overview of the available options
- `-i INFILE` loads the input file in the input format as described above
- `-cmd CMDFILE` loads the command file in the output format as described above
- `-log LOGFILE` specifies the file into which to dump information about the simulation. This is optional, as the same information will always be logged in standard output.