NOI '13 P3 - Little Q's Training

Time limit: 4.5s **Memory limit:** 256M

National Olympiad in Informatics, China, 2013

Little Q recently discovered a new video game. The objective of the game is to train from a lowly noob to a powerful master.

Facing an intricate virtual world, little Q must make careful decisions about the events that lie ahead. For example, whether he should accept a stranger's challenge to battle, whether he should accept or reject an offer to trade his precious blade for another's martial arts manual, etc. Furthermore, Little Q's every decision might impact how his future develops: for example, voluntarily facing a master may lead to heavy losses, but not accepting the fight may lead to never being able to see the master again.

Little Q has played through this game many times, yet still he has not been able to play through to his desired ending. So, he has painstakingly tracked down the game's storyboard. Surprisingly, the storyboard for this game does not at all resemble the average video game storyboard. Instead, it very much resembles source code. The storyboard is outlined as follows:

- **Value**: Can be one of two types – a constant or a variable.
- **Constant**: An integer.
- **Variable**: An integer, initially 0, that can change throughout the game. Each variable is identified by its unique positive integer index.
- **Event**: The entire storyline consists of some number of events. All events are numbered sequentially starting from 1. Can be one of three types – a normal event, an optional jump, or a conditional jump.
- **Position**: An integer representing the number of the next event that will happen. If an event with this number doesn't exist, then the game has found an ending and will stop. Initially, the position will be 1.
- **Normal event**: A variable will increase or decrease by a value. Afterwards, the position will be incremented by 1.
- **Optional jump**: Two integers. Reaching here, the player must pick one of the two integers. Afterwards, the position will be set equal to the selected integer.
- **Conditional jump**: Two values and two integers. Reaching here, if the first value is smaller than the second value, then the position will be set equal to the first integer. Otherwise, the position will be set equal to the second integer.

Little Q believes that the objective of the game is to make the variable called `EXP` (index 1) as large as possible.

Input Specification

There will be 10 files `train1.in` to `train10.in` that will be given to your program (through standard input). They can be downloaded here for you to study: train.zip.

For each test case:

The first line of input will contain an integer from 1 to 10, representing the test case number. Test case `traini.in` will have i on its first line.
The second line of input will contain two positive integers n and m, representing the number of events and the number of variables.
The following n lines each describes an event. These events are numbered sequentially from 1 to n.
The format of each event is outlined as follows:

Type	Format	Example
Constant	`c <Integer>`	`c -2`
Variable	`v <Positive Integer>`	`v 5`
Normal event	`<Variable> + <Value>` `<Variable> - <Value>`	`v 1 + c -1` `v 2 - v 2`
Optional jump	`s <Integer 1> <Integer 2>`	`s 10 20`
Conditional jump	`i <Value 1> <Value 2> <Integer 1> <Integer 2>`	`i c 99 v 2 0 1`

Output Specification

For each test case, output some number of lines. On each line, output a single character, either `1` or `2`, representing the decisions made for optional jumps as the game progresses. The number of lines outputted must strictly match the number of optional jumps encountered throughout the gameplay.

Sample Input

```
0
11 2
v 2 + c 19
i v 2 c 3 7 3
s 4 7
v 1 + c 13
v 2 - c 3
i c 0 c 1 2 0
i v 2 c 5 12 8
s 9 12
v 1 + c 23
v 2 - c 5
i c 0 c 1 7 0
```

Sample Output

```
1
1
1
2
1
1
```

Explanation

If the storyboard was numbered as follows,

```
1
2
3
4
5
6
7
8
9
10
11
```

```
v 2 + c 19
i v 2 c 3 7 3
s 4 7
v 1 + c 13
v 2 - c 3
i c 0 c 1 2 0
i v 2 c 5 12 8
s 9 12
v 1 + c 23
v 2 - c 5
i c 0 c 1 7 0
```

then according to the sample output, the positions would undertake the following values:

$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 2 \to 3 \to 4 \to 5 \to 6 \to 2 \to 3 \to 4 \to 5 \to 6 \to 2 \to 3 \to 7 \to 8 \to 9 \to 10 \to 11 \to 7 \to 8 \to 9 \to 10 \to 11 \to 7 \to 12$

When the position becomes 12, the game ends. The final value of variable 1 is 85.
Event 1 consists of variable 2 increasing by 19. We can think of it as receiving 19 units of initial gold.
Event 6 is an unconditional jump to event 2. It is not hard to see that this is a cycle. From event 2 and event 3, we can observe that if variable 2 is less than 3 (gold is not enough to make purchase) or if a choice is made to exit, then the cycle will be broken. Within the cycle, events 4 and 5 will cost 3 gold units to obtain 13 EXP.
Events 7 and 11 form a similar loop only with different parameters. It costs 5 units of gold to obtain 23 EXP.
Noticeably, the sample is an unbounded knapsack problem. The sample output yields its optimal solution.

Grading

For each test case, the following method will be used to determine your score out of 10:
If your output is invalid, then you will score 0 points.

If your output executes more than 10^6 lines of storyboard, then you will score 0 points.
If your output allows the story to terminate normally, then you will score 1 point.
If your output allows the story to terminate normally, and your EXP at the end is a positive value, then you will score 2 points.
We have set up 8 parameters $a_3, a_4, \ldots, a_{10}$.
If your output allows the story to terminate normally, and your EXP at the end is no less than a_S, then you will score S points.
If multiple of the above conditions are satisfied, the one with the greatest score will be counted.
Formally speaking, if your solution is valid and yields a final value of v_1 for variable 1, then your score will be given in accordance with the following table.

Score	Condition
10	$v_1 \geq a_{10}$
9	$a_9 \leq v_1 < a_{10}$
8	$a_8 \leq v_1 < a_9$
7	$a_7 \leq v_1 < a_8$
6	$a_6 \leq v_1 < a_7$
5	$a_5 \leq v_1 < a_6$
4	$a_4 \leq v_1 < a_5$
3	$a_3 \leq v_1 < a_4$
2	$0 < v_1 < a_3$
1	$v_1 \leq 0$

Experimentation

We provide a tool train_check.py for you to help check if your output program is valid. The usage for this program is:

`train_check.py <case_no>`

where `<case_no>` is the number of the test case. For example:

`train_check.py 3`

will test `train3.out` to determine whether it is accepted.

After you invoke this program, the checker will provide the result to the execution of your output file in one of the following ways:

- `Abnormal termination`: An unknown error occurred.
- `Input/Output file does not exist.`: We cannot load your input or output file.
- `Output invalid.`: There is an error with your output file. A general error message may now be included.
- `Correct! Your answer is x.`: Your output is acceptable. The final value of the EXP variable is x.

Extra Features

The checker program can also be used on any input and output file. The usage is as follows:

`train_check.py <input_file_name> <output_file_name>`

where `<input_file_name>` and `<output_file_name>` are the input and output file names respectively. For example:

`train_check.py train3.in train3.out`

will check if `train3.out` is accepted.

Using `-w` will output the results of execution at each step. The usage is:

`train_check.py -w <input_file_name> <output_file_name>`

or:

```
train_check.py -w <case_no>
```

e.g.:

```
train_check.py -w train3.in train3.out
```

Problem translated to English by **Alex**.