Twelfth Quinnipiac University Programming Competition for High School Students

Quinnipiac University

April 29, 2017

Instructions

1. Program submissions are done using the Programming Contest Control System (PC^2). See the separate sheet for instructions on submitting your programs.
2. You may assume the input is formatted as shown in the problem, and your program must work on such input. You may assume there are no tabs in the input or trailing spaces.
3. Your output must match the specified output *exactly*. This includes not having any trailing spaces.
4. If your program crashes, produces incorrect output, or times out, no information will be given as to the input that caused the problem.

1 Rock Climbing

Boomer[1] loves to take pictures of the Quinnipiac campus and the surrounding neighborhoods for the Quinnipiac website. For his first trip of the day, he decides to climb up and down the head of the Sleeping Giant. Of course, he is never happy with any single shot so after every photo he either climbs up or down a few feet. Your task is to track the climbing that Boomer does on his mission to capture the best aerial shot, keeping track of his highest point, ending height, and total climbing.
Boomer always starts his mission at 0 feet, never climbs below 0 feet, and can climb as high as he wants - though none of the inputs has him going over 100000 feet. So suppose Boomer's plan is to climb 100 feet, descend 200 feet, and climb 20 feet for his shots. He will start at 0 feet, climb to 100 feet, *attempt* to descend 200 feet but get stopped by the ground, and then climb 20 feet. Thus, after completing his task, Boomer will have reached a highest peak of 100 feet, ended at 20 feet, and climbed a total of 120 feet.

Input

The first line contains the number N of cases with $0 < N \leq 100$. This is then followed by N lines, one line per case. Each line starts with the number of photo stops S, $0 < S \leq 100$ that he makes, followed by S climbs C (in feet) (positive numbers mean ascending, while negative numbers mean descending) with $-1000 \leq C \leq 1000$. All values are integers.

Output

The output consists of one line for each case, where the line contains three numbers separated by spaces: the highest point reached, the ending height, and the total climbing (asending only).

Input

```
3
3 100 -200 20
7 10 -5 8 -20 60 -10 -4
8 12 23 84 -50 -3 6 0 -10
```

Output

```
100 20 120
60 46 78
119 62 125
```

[1] Quinnipiac's hard-working mascot

2 The Gift Shop

After earning money from selling his wonderful aerial photos of campus, Boomer heads to the Gift Shop to buy some merchandise. At the store, he picks out various quantities of items each with a different price. Help Boomer decide if he has enough money to buy his wish list.

Input

The first line contains the number N of cases with $0 < N \leq 100$. This is then followed by N lines, one line per case. Each line starts with a number of items S, $0 < S \leq 100$, which is then followed by S pairs of values, where the first value in a pair is the quantity Q for that item and the second value is the price P of that item with $0 < Q \leq 100$ and $0 < P \leq 100.00$. The price P is specified in dollars and possibly cents. These values are then followed by the amount of cash C that Boomer earned from his photos with $0 < C \leq 1000000.00$.

Output

The output consists of one line for each case, where the line is one of the following phrases:

- `BOOMER HAS THE EXACT AMOUNT HE NEEDS`
- `BOOMER HAS MORE THAN ENOUGH`
- `BOOMER NEEDS MORE MONEY`

Input

```
3
4 3 0.10 5 2.40 6 2.10 8 3.00 80.56
4 1 1.11 2 2.20 3 3.30 4 4.40 33
2 10 0.01 4 0.05 0.30
```

Output

```
BOOMER HAS MORE THAN ENOUGH
BOOMER NEEDS MORE MONEY
BOOMER HAS THE EXACT AMOUNT HE NEEDS
```

3 Letter Scramble

At the last hockey game of the season, Boomer gave several students a separate letter to hold up high so that they would spell the word "BOBCAT." Unfortunately, they did not sit in the correct order so he had to provide them with instructions to move about and unscramble the letters. Boomer, in his infinite wisdom, decided to make a technique out of it so he could use it for future games and messages as well. Here are the commands he created:

- `swap I J`: The letters at index I and index J are swapped with counting starting at 0. The indices are guaranteed to reference specific letters in the message.
  For example, `swap 0 3` would transform the string `TAOBBC` to `BAOTBC`.
- `rotate K`: Rotates the string K steps to the left (negative values rotate right) with $-1000 \leq K \leq 1000$. For example, `rotate 2` would transform the string `BAOTBC` to `OTBCBA`.
  And, `rotate -10` would transform the string `OTBCBA` to `BCBAOT`.
- `reverse I J`: The span of letters from index I to index J (inclusive of both) should be reversed. As before, counting starts at 0 and the indices are guaranteed to reference specific letters in the message and $I \leq J$. For example, `reverse 1 3` would transform the string `BCBAOT` to `BABCOT`.
- `move I J`: Moves the letter at index I to index J by removing the letter at I and inserting it so it ends up at J. Again, counting starts at 0 and indices are guaranteed to reference specific letters in the message. For example, `move 1 4` would transform the string `BABCOT` to `BBCOAT`.
  And, `move 3 1` would transform the string `BBCOAT` to `BOBCAT`.

Input

The first line contains the number N of cases with $0 < N \leq 100$. This is then followed by N cases. The first line of each case starts with the starting message M where M has length more than 0 and no more than 100 and each character in M is a letter in A-Z (inclusive). The next line of each case is the number S of instructions to perform with $0 < S \leq 100$. This is followed by S lines, with each line being an instruction of the form described above.

Output

The output consists of one line per case, where the line is the message after performing the instructions.

Input

```
3
TAOBBC
6
swap 0 3
rotate 2
rotate -10
reverse 1 3
move 1 4
move 3 1
BOBCAT
5
move 1 3
move 4 1
reverse 1 3
rotate 26
swap 0 3
BDGJOOO
6
reverse 0 6
reverse 3 6
swap 6 5
move 3 5
rotate -36
move 1 5
```

Output

```
BOBCAT
TAOBBC
GOODJOB
```

4 Back to the Gift Shop

Boomer decides to head back to the gift shop after earning money from his latest gig. This time after picking out the various quantities of items with differing prices, he wants to see what is the maximum number of items he can buy with his money.
Suppose that he identified three $5 T-shirts, four $3 hats, and eight $1.50 binders. If he had $23.50 of his hard-earned money, he could buy at most 11 items: eight binders, two hats, and a T-shirt.

Input

The first line contains the number N of cases with $0 < N \leq 100$. This is then followed by N lines, one line per case. Each line starts with a number of items S, $0 < S \leq 100$, which is then followed by S pairs of values, where the first value in a pair is the quantity Q for that item and the second value is the price P of that item with $0 < Q \leq 100$ and $0 < P \leq 100.00$. The price P is specified in dollars and possibly cents. These values are then followed by the amount of cash C that Boomer earned from his last gig with $0 < C \leq 1000000.00$.

Output

The output consists of one line for each case, where the line is simply the maximum number of items that he could purchase.

Input

```
3
3 3 5.00 4 3.00 8 1.50 23.50
3 8 22.50 7 14.50 6 33.33 10.00
4 4 14.00 5 11.00 6 12.00 4 10.00 83.99
```

Output

```
11
0
7
```

5 Stadium Seating

Boomer decided to help the Athletic Department out with stadium seating. After spending considerable time at the massive arena testing out every one of the arena's K seats, he came to his own personal ranking of every single seat, 1 being the absolute best seat in the arena and N being the least preferred seat. At any given point, he has a list of possibly overlapping segments of occupied seats and now he wishes to have a system that identifies the best (lowest ranked) unused seat.
For example, he might have that seats $1-10$, $15-20$, $25-30$, and $8-17$ are all occupied. Noting that the last segment $8-17$ overlaps two other segments, the best available seat is seat number 21. This arena is *massive* the maximum number of seats is $K = 2,000,000,000$.

Input

The first line contains the number N of cases with $0 < N \leq 100$. This is then followed by N lines, one line per case. Each line starts with a number of segments S, $0 < S \leq 100$ that are stored, followed by S intervals of the form $A - B$ where A represents the start of the segment and B represents the end of the segment. In all cases, $0 < A \leq B \leq K$. All values are integers.

Output

The output consists of one line for each case, where the line is simply the number of the first available seat.

Input

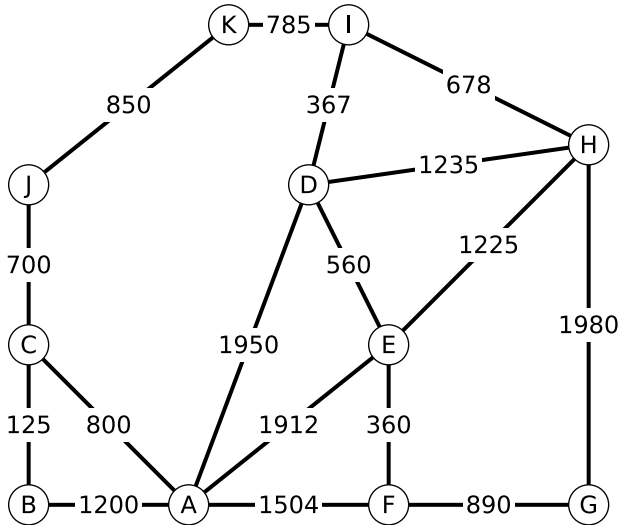
```
3
4 1-10 15-20 25-30 8-17
5 21-40 91-100 35-89 1-7 8-20
3 1-3 4-1234567890 1324567890-1324567892
```

Output

```
21
90
1234567891
```

6 Hiking the Giant

Boomer was given another photo shot opportunity along the trails of the Sleeping Giant. This time he was given a map (below) along with instructions on where to go for each shot. The eleven possible locations of the shots are indicated by letters A-K. The distances and possible paths between the various locations are indicated with integral values. The instructions, which always have Boomer starting at location A, are a sequence of numbers indicating which location to go to next. A positive number n means count n paths clockwise from the path currently used while a negative number n means count n paths counterclockwise. For example, if Boomer had arrived at location D from location E, then the number 1 would mean take the path from D to A while the number -1 would mean take the path from D to H. The number 0 means go back along the path most recently taken, from D to E in this case. The numbers can wrap around so the number 6 would indicate going from D to I. In the first step, the number will always be in the range 1 to 5 with a 1 indicating going from A to B and 5 from A to F. Your task is to determine the order of locations that Boomer visits given his instructions along with the total distance (in feet) traveled.
For example, if the instructions were listed as `3 2 -5 6 3`, Boomer would start by going from A to D, then to H, then to E, then back to A, and finally finishing at C. This yields a total distance of $1950+1235+1225+1912+800 = 7122$.



Input

The first line contains the number N of cases with $0 < N \leq 100$. This is then followed by N lines, one line per case. Each line consists of a number S, $0 < S \leq 100$ followed by S turning directions D, $-100 \leq D \leq 100$. All values are integers.

Output

The output consists of one line for each case, where the line contains the list of locations Boomer visited separated by spaces, followed by the total distance traveled by Boomer.

Input

```
3
5 3 2 -5 6 3
9 1 1 1 1 1 1 1 1 1
9 5 -2 7 -23 0 1 0 0 -1
```

Output

```
A D H E A C 7122
A B C J K I H G F A 8712
A F E H D H I H I K 8378
```

7 Last Trip to Gift Shop

Boomer decides to head back to the gift shop for one last purchase. This time he has a gift card that he received from his fans. Since this particular card can only be used once and no cash is returned for any unused portion, Boomer really wants to spend as much as he possibly can, without going over the limit. Devise a program that can help Boomer maximize his spending after picking out various quantities of items with different prices. Notice he does not want more than the quantity he chose but does want to spend as much as possible within the limits of his gift card.
Suppose that he identified three $5 T-shirts, four $3 hats, and eight $1.50 binders. If his gift card was worth $25.25, he could buy two T-shirts, three hats, and four binders, spending a total of $25 and leaving him with exactly $0.25 left-over.

Input

The first line contains the number N of cases with $0 < N \leq 100$. This is then followed by N lines, one line per case. Each line starts with a number of items S, $0 < S \leq 100$, which is then followed by S pairs of values, where the first value in a pair is the quantity Q for that item and the second value is the price P of that item with $0 < Q \leq 30$ and $0 < P \leq 100.00$. The price P is specified in dollars and possibly cents. These values are then followed by the amount C on Boomer's gift card with $0 < C \leq 100.00$.

Output

The output consists of one line for each case, where the line is simply the maximum amount of money that he could spend without going over his limit. The format should be in dollars and cents (as indicated below).

Input

```
5
3 3 5.00 4 3.00 8 1.50 25.25
4 3 2.10 4 0.20 5 3.50 4 8.25 8.44
3 2 1.00 3 3.00 4 0.10 100.00
3 3 5.00 4 3.00 8 1.50 23.50
3 8 22.50 7 14.50 6 33.33 10.00
```

Output

```
25.00
8.30
11.40
23.50
0.00
```

8 Assemboomer

After realizing he wanted to buy far more gear at the QU gift shop than he can afford just taking photos, Boomer decided he wanted to start coding complex projects but he was not thrilled with the existing programming languages. So, he devised his own, which he called Assemboomer. (It is a work in progress.) Currently, the system uses four registers labelled `a`, `b`, `c`, and `d` (initially all reset to 0) and supports various commands. Execution starts at the first line and continues sequentially until the `end` command is reached or execution goes beyond the last line of code. The exception to executing sequentially is after a jump (`jnz`) command, where execution can continue to the instruction indicated by the command. Boomer, oddly, decides that his numbers can only be in the range -2017 to 2017 (inclusive of both)[2] with math operations wrapping around. For example, $2017 + 1 = -2017$ and $-2017 - 1 = 2017$ and $2017 + 3 = -2015$. His language currently supports the following instructions:

- `reset`: Sets all of the registers to 0.
- `print`: Prints out the values of each register on a single line separated by spaces.
- `swap R S`: Swaps registers R and S (each being a letter `a`, `b`, `c`, or `d` where they could reference the same register.)
- `load R V`: Load the value V into register R. V could be a register (`a` to `d`) *or* a numerical value $-2017 \leq V \leq 2017$.
- `add R V`: Add the value of V to register R. V and R are defined the same as above.
- `neg R`: Set register R to its negative (e.g. 5 becomes -5, -10 becomes 10, 0 stays 0).
- `jnz R V`: If the value in register R is not zero, execution jumps V instructions forward (if positive) or back (if negative). V and R are again defined the same as above. If either the value in register R is zero *or* the value of V is zero, execution continues as normal. Instruction jumping is relative to the current instruction. If the new location after a jump is *before* the first line of instruction, code starts again at the first line. If the new location after a jump is *after* the last line of instruction, the program ends. For example, suppose register `a` has value 5 and register `b` has value 3 and the current instruction is on line 7. Then `jnz a b` would jump (because 5 is not zero) 3 steps to line 10. And line 10 would be the next line executed - unless there was no line 10, in which case the program would end.
- `end`: Whenever this statement is executed, the program terminates immediately.

Although Boomer has created his language, he does not yet have an interpreter to run programs written in his language. Help Boomer out by creating an interpreter for his language.

Input

The input is a single "program." The first line contains the number N of lines in the program with $0 < N \leq 100$. This is then followed by N lines of instructions as described above.

Output

The output is simply the output of the program until it ends. These are just the various prints made by his program. (We guarantee that the program provided will terminate at some point.)
See the next page for a sample "program" and output.

[2] He reasoned that he could add one more to that range every year.

Input

```
25
jnz d 13
load c 10
load d 2
neg d
load a 200
print
load b a
add a b
add c d
jnz c -4
swap a b
print
jnz a -100
reset
load a 300
load b 8
print
add c a
add b -1
jnz b -3
swap a c
print
end
load a -100
print
```

Output

```
200 0 10 -2
400 200 8 -2
800 400 6 -2
1600 800 4 -2
-835 1600 2 -2
-835 -1670 0 -2
300 8 0 0
300 7 300 0
300 6 600 0
300 5 900 0
300 4 1200 0
300 3 1500 0
300 2 1800 0
300 1 -1935 0
-1635 0 300 0
```