NINTH ANNUAL JUILFS CONTEST
SPRING 2017



Department of Computer and Information Science
University of Oregon
2017 June 03

contributors: Jim Allen, Ryan Collier, Chuanqi Lan, Joseph McLaughlin, Jordan Merckling, Chris Wilson

A. William Got Fired [10 points]

William works as a bookkeeper at a factory that produces calculators. His job requires he create arithmetic expressions for various important bookkeeping activities. William is often annoyed with having to manage the order of operations and parentheses in his expressions and often wonders if there's a better way.

One day when eating lunch, William overhears two software engineers discussing the benefits of postfix notation and how it doesn't require parentheses. Unfortunately, all William hears about this form of notation is that operators are always written after operands. William returns from lunch and begins recording expressions with this in mind.

On one page William records the next operand in his expression and on another the writes next operator. To save space, if there exists multiple sequential operators William records that operator and the number of times to apply it sequentially. William evaluates his expressions by evaluating the leftmost operations first, applying the leftmost operator to the leftmost operands.

Example:

```
1 2 5 4 3 6 8
+3-/*
```

```
((1+2+5+4-3)/6)*8 = 12
```

After a week of this practice, William was fired. Your team was hired by William's previous employer to write a program that evaluates William's notation.

Input

The first line of input will contain a single integer **T**, followed by **T** expressions. Each expression consists of three lines, the first consisting of an integer **N**, the second contains **N** space separated integers, and the third contains instructions for **N-1** operations `(+|-|*|/)[2-9]?`.

Output

For each expression print the value that the expression evaluates to. All expressions will evaluate to integers.

Sample Input	Sample Output
2 7 1 2 3 4 6 5 4 +-3/* 14 10 5 23 3 9 3 49 19 5 23 24 2 3 5 +2*-/+-/+2/3	-8 2

B. Point in Circle (8 points)

You are given a circle in a coordinate system, with its radius and the coordinate of the center. You are also given many points and their locations. You are asked to find if each point is in the circle or on the circle or outside the circle.

Input

The first line of input contains three space-separated integers, x, y, r and m. Which (x, y) is the coordinate of the center and r is the radius of the circle, and m indicates m line after this line(means there are m points).

Each of the next m lines contains two space-separated integers, a, b which indicates the coordinate of the point.

Output

Print m lines, with “Yes”, “No”, or “On” indicating the point is in the circle, outside the circle or on the circle.

Sample Input	Sample Output
0 0 5 6 1 1 2 2 6 6 5 1 3 5 5 0	Yes Yes No No No On

C. Mixed Up Capital Letters

(9 points)



Everyone likes to post a good emotional rant online. The best way to show the depth of emotion is to mix up capital letters with lower case letters in a random way. You are to write a program to take a phrase and see if it meets some basic criteria of mixing up the cases of the characters.

Here a phrase is a string consisting of up to 140 characters – these will be upper case letters [A-Z], lower case letters [a-z], and spaces (the first and last character will be a letter not a space). To have a "GooD mIx" you must have an equal number of upper and lower case letters (spaces do not count) – if the number of letters is odd then they can differ by at most one. In addition, the first and last letters must be a different case – for example, "NopE" is not good since the first and last letters are both upper case.

The input will be a number n, $1 \le n \le 100$, followed by n phrases of at most 140 characters. The output should be a statement for each phrase indicating "GooD mIx" or "BAD Mix", depending on whether the phrase meets the criteria above.

For example, on the input

```
4
I aM A GooD TeEchEr
NopE
NOT GOOD Enuf
yeSS
```

the desired output is

```
GooD mIx
BAD MIx
BAD MIx
GooD mIx
```

D. Bugs Bunny Binary (11 points)

Decimal	Binary	1-bit Reflected Binary	2-bit Reflected Binary	3-bit Reflected Binary	4-bit Reflected Binary
0	0	0	00	000	0000
1	1	1	01	001	0001
2	10		11	011	0011
3	11		10	010	0010
4	100			110	0110
5	101			111	0111
6	110			101	0101
7	111			100	0100
8	1000				1100
9	1001				1101
10	1010				1111
11	1011				1110
12	1100				1010
13	1101				1011
14	1110				1001
15	1111				1000

Reflected binary encoding was invented by Frank Gray in 1953 for use in color televisions. The problem was that adding 1 to a number in standard binary may cause any number of bit flips. With the electromechanical tubes in use at the time, the individual bits may switch at slightly different times, causing superfluous states. In reflected binary code, each increment involves flipping exactly one bit. The set of n-bit codes are created from the (n-1)-bit codes by prepending a zero to the codes in order and then prepreding a one to the codes in reverse order, hence "reflected binary".

From the table, you can see that converting between a standard binary expansion $b_0b_1b_2\ldots b_n$ and a reflected binary expansion $g_0g_1g_2\ldots g_n$ involves using *exclusive or* ($\oplus$) in the bitwise functions

$$g_i = \begin{cases} b_i & i = 0 \\ b_i \oplus b_{i-1} & i > 0 \end{cases} \qquad b_i = \begin{cases} g_i & i = 0 \\ g_i \oplus b_{i-1} & i > 0 \end{cases}$$

Notice that in both functions, the bits are numbered from left to right, rather than the more conventional right to left.

In this problem, the first line of input will be a single number n where $0 < n \leq 100$, followed by n lines containing exactly one reflected binary expansion of less than 32 bits. For output, you should have n lines of decimal numbers, decoding the input.

Sample Input	Sample Output
5 000000 0001 1010 011001 1101100110011100010101101101000	0 1 12 17 1217121712

E. When Change Changes

(11 points)



You are a convenience store cashier on the strange planet of Trollmeme. After people pay you for the bizarre objects they purchase you must give them back their change. The change number pops up on your register till, given an infinite amount of coins what is the minimum number of coins you can give the customer back to get them the correct amount of change due. One catch is that Trollmeme has a very unstable economy, so the value of coins is constantly changing. The change amounts can update as often as every transaction. Luckily, Trollmeme's government has insisted that all coin sets remain canonical, this means always giving them the largest coin possible from the change remaining will lead to the minimum number of coins back.

Your task is to write a program that calculates the minimum number of coins that must be given to a customer with the current coin set. The input will be a number n, $1 \leq n \leq 20$, which will represent the number of transactions, the next lines will be the change due for that transaction c, $c < 100$, then the number of coins in that transactions set s, $1 \leq s \leq 5$, then the last line for each transaction will be the s coins in the set, each set starts with 1 then has s-1 (sorted from smallest to largest) more canonical coins (no coin will exceed a value of 99).

Example Input:

```
2
20
4
1 2 4 8
42
5
1 5 10 25 50
```

Expected Output:

```
3
5
```

F. A Graceful Grace Period



(12 points)

A teacher wants the best from his students, but doesn't want to be taken advantage of. So, he devised a plan to deal with late assignments. If an assignment is late it is only worth 50% of the total earned. But, each student is given a number of "passes" that are each worth two extra days to work on a single assignment. All late fines (50% mark down) and passes are applied at the end of the term. As the teacher's assistant, it is your job to find the best assignments to apply the passes to. Please try to give each student the best grade possible!

The input will be the number of students n, $1 \le n \le 30$, the number of assignments a, $1 \le a \le 20$, and the number of passes given out p, $1 \le p \le 5$. These numbers will be followed by each student's grade for every assignment g, $0 \le g \le 100$ and how many days late that assignment was L, $0 \le L \le 20$. Passes can be stacked to give a single assignment two more days (2 * pass) after the due date to get full credit. As in the example input, you would want to apply both passes to the assignment that achieved 100% so the student received a score of 78 instead of 74. The output should be a student's number followed by their best possible grade. Which would be an average of their assignment scores. It should not be assumed that any assignments were late. Whole numbers only, please. If rounding, always round down.

For example, the input:

2
3
2
100 3
94 0
80 1
100 0
96 1
88 10

Describes two students given three assignments, two late passes and how well they did on those assignments. The desired output is the student's number followed by their final grade:

1 78
2 80