IP OR NOT

Points: 50

Your boss asks you to write a program that checks whether a string entered by the user is a valid IP address or not.

Note: Strings are valid IP addresses if they are in the format X.X.X.X where X is an integer varying from 0 to 255

Input

The input is a file with each line containing a string

Output

The output of the program shall indicate, for each line, whether the given string is “Valid” or “Invalid” IP address.

Sample Input

172.16.32.22
155.555.23.23
Hello world
192.168.100.100:8090

Sample Output

Valid
Invalid
Invalid
Invalid

Indra Pal Singh

TRIANGLE NUMBERS
Points: 50

The sequence of triangle numbers is generated by adding the natural numbers. So the 7^(th) triangle number would be 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28. The first ten terms would be:

1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...

Let us list the factors of the first seven triangle numbers:

1: 1
3: 1,3
6: 1,2,3,6
10: 1,2,5,10
15: 1,3,5,15
21: 1,3,7,21
28: 1,2,4,7,14,28

We can see that 28 is the first triangle number to have over five divisors.

Find the value of first triangle T number to have over N divisors
N ranges [1,500]
T ranges [1,1000000000]

INPUT: number N in K lines
OUTPUT: number T in corresponding K lines

SAMPLE INPUT
20
36
23
102
346
448

SAMPLE OUTPUT
630
3240
3570
73920
103672800

Aditya Ranjan

CAKE PIECES
Points: 75

We all celebrated our last birthday here in BIT. On this ocassion my friend Sanjeev was presented with a special cake . When viewed from above, the cake is a square, with an empty square hole inside. Both squares are centered at (0, 0) and their sides are parallel to the x- and y-axes.

Sanjeev cut the cake using several horizontal and vertical cuts. These cuts are given in the int[]s horizontalCuts and verticalCuts. The i-th horizontal cut is a line parallel to the x-axis which goes through the point (0, horizontalCuts[i]). Likewise, the i-th vertical cut is a line parallel to the y-axis which goes through the point (verticalCuts[i], 0). All cuts have infinite lengths.

You are given an int cakeLength, half of the side length of the outer square, and an int holeLength, half of the side length of the inner square hole. Note that both of these numbers are halves of the sides of the corresponding squares. Return the number of pieces of cake that will exist after all the cuts are performed.

Input

Each test case hase cakeLength on the first line followed by the hole Length in the next line. The next line contains the number of horizontal cuts followed by the elements of the array horizontalCuts. The next line contains the number of vertical cuts followed by the elements of the array verticalCuts.

Output

The output should give the number of pieces of cake.

Sample Input

```
5
3
2 1 -4
1 1
```

Sample Output

```
6
```

Sushant

LUCKY SUM
Points: 75

A friend of mine thinks 4 and 7 are lucky digits, and all other digits are not lucky. A lucky number is a number that contains only lucky digits in decimal notation.

Some numbers can be represented as a sum of only lucky numbers. Given a number N, you have to find the minimum number of lucky numbers that sum up to the given number.

Input

Each test case contains the number N(<=1000000) on a separate line.

Output

The output should give minimum number of lucky numbers that sum up to the given number.Print "NOT POSSIBLE" in case the sum is not possible.

Sample Input

```
4
13
12
```

Sample Output

```
1
NOT POSSIBLE
3
```

Sushant

TRIVIA GAME

Points: 100

You and your friends got together for a Trivia night at the local pub. Each question is worth a number of points; the i-th element of points corresponds to the score received if you correctly answer the i-th question, but you lose that many points if you answer it incorrectly. The questions are given in the order specified in points, and you must answer each question before the next is asked.

In addition, after each correct answer you will receive a token, and you keep all of your tokens if you answer a question incorrectly. If you then have tokensNeeded tokens, the pub will immediately take all of your tokens and award you additional bonus points. The bonuses are different for each question; element i of bonuses corresponds to the bonus you receive if you win the bonus on question i.

You know the answer to all the questions and want to maximize the number of points that you receive. Return the maximum points that you can receive if you correctly choose which questions to answer.

Input

Each test case starts with the number of questions , N(<=50). Next line contains the points alotted to each question. Next line contains tokensNeeded. Last line contains the bonuses for all the questions.

Output

The output should give the maximum points that you can receive if you correctly choose which questions to answer.

Sample Input

```
6
1 2 3 4 5 6
3
4 4 4 4 4 4
6
1 2 3 4 5 6
3
1 1 1 20 1 1
```

Sample Output

```
19
39
```

Sushant

TEMPORARY VARIABLES

Points: 150

The simple programming language ASSIGN allows only assignment statements (one statement per line) with the following format. Each assignment statement consists of a Left Hand Side (called LHS) and a Right Hand Side (called RHS) which are separated by the = sign with at least one space before and after the =.

- An LHS can be
  - A (regular) variable name consisting of letters which may be small or capital. Digits are not allowed in variable names.
  - A temporary variable name beginning with an _t followed by a number as shown in the examples below.
- A RHS can be complex
  - An infix binary expression consisting of two operands (defined below) and one of the following binary operators: +, *, -, /. The operator and operands are separated by a space.
  - A prefix unary expression consisting of one operand and one of the following unary operators: &, *, -. The operator and operand is separated by a space.

  Or an RHS can be simple
  - A single operand without any operator

An operand can be either a variable name, or a temporary name or an integer constant in decimal format.

You have to write a program to read in a sequence of assignment statements and using the rules explained below until they no longer apply, generate the final output sequence. Note that there will always be only one possible output.

- Rule 0: At any point in the derivation, all temporary variables names in the output must be in order. This means that _t2, for example, should not appear as an LHS before the first occurrence of both _t0 and _t1 as an LHS.
- Rule 1: If the RHS is complex and if the LHS of an assignment is not a temporary variable, insert a new temporary variable as shown below.

Input		Output
value = value * change newvalue = value	=>	_t0 = value * change value = _t0 newvalue = value

- Rule 2: If the RHS of two statements are identical complex expressions, but the corresponding LHSs are different temporary variables, then eliminate the second temporary variable using the first one as shown below.

Input		Output
_t1 = a + b		_t0 = a + b
_t2 = c + _t1		_t1 = c + _t0
d = _t2	=>	d = _t1
_t5 = a + b		_t0 = a + b
_t6 = c + _t5		_t1 = c + _t0
e = _t6		e = _t1

Note that Rules 0 and 1 have also been used here. For example, the expression ``c + _t1'' in the input becomes ``c +_t0'' in the output and since temporary variable ``_t5'' of the input is same as ``_t0'' if the output, expression ``c + _t5'' in the input also become ``c + _t0'' in the output.

- Rule 3: If the RHS of the assignment is a complex binary expression, but contains no variables, then replace RHS expression by its value, and replace the LHS by a new temporary variable as shown below.

Input		Output
a = 10 * 20	=>	_t0 = 200
		a = _t0

Input

The input will contain several test cases. The first line contains a positive integer N giving the number of test cases. For every test case the first line will a number m giving the number of assignment statements in the input program. This will be followed by m lines, one assignment statement per line, corresponding to the input program.

Output

The output for each test case must be the sequence obtained using the rules explained above as long as any of them is applicable. The outputs for consecutive test cases should be on consecutive lines.

Sample Input

```
2
1
a = 10 * 20
5
a = a * b
b = * a
c = - a
```

```
_t13 = * a
b = & _t13
```

Sample Output

```
_t0 = 200
a = _t0
_t0 = a * b
a = _t0
_t1 = * a
b = _t1
_t2 = - a
c = _t2
_t1 = * a
_t3 = & _t1
b = _t3
```

Indra Pal Singh