2022 ACM Programming Contest

- Complete as many questions as you can during the contest.
- **Only one computer may be used per team.**
- No connection to the Internet may be made during the contest.
- All questions require you to read the test data from standard input and write results to standard output; you cannot use any non-contest files for input or output.
- The input to problems will consist of multiple test cases unless otherwise noted.
- Programming style is not considered in this contest. You are free to code in whatever style you prefer. Commenting code is not required.
- All communication with the judges will be handled by the PC2 environment.
- Allowed programming languages: C, C++, Java, and Python.
- **Java users: remember to call `nextLine()` on a scanner after reading in an integer when needed.**
- All programs will be re-compiled prior to testing with the judges' data.
- Output must match the sample format exactly. Do not print prompts unless asked to.
- Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and string libraries are allowed. The standard Java API is available and can be used, except for those packages that are deemed dangerous by contest officials (e.g., that might generate a security violation)
- Hardcopy reference materials are allowed, but not electronic material. Calculators are fine, but no laptops, phones, USB drives, etc. are allowed.
- **Netbeans users: Remove package statement before you submit**

Judges decisions are to be considered final. No cheating will be tolerated.

Problem A - Binarize It

Professor Boolando can only think in binary, or more specifically, in powers of 2. He converts any number you give him to the smallest power of 2 that is equal to or greater than your number. For example, if you give him 5, he converts it to 8; if you give him 100, he converts it to 128; if you give him 512, he converts it to 512.

Input The first input line contains a positive integer, n, indicating the number of values to binarize. The values are on the following n input lines, one per line. Each input will contain an integer between 2 and 100,000 (inclusive).

Output At the beginning of each test case, output "Input value: v" where v is the input value. Then, on the next output line, print the binarized version. Leave a blank line after the output for each test case.

Sample Input

```
3
900
16
4000
```

Sample Output

```
Input value: 900
1024

Input value: 16
16

Input value: 4000
4096
```

Problem B - Gathering Gamers

A line of new students is waiting to get their WrightOne cards. Some of these students are gamers (as evident by their t-shirts), and they want to meet other people who share their passion. If a gamer spots another gamer that is within d spots of them in line, the gamer closer to the beginning of the line will move to just in front of the other gamer so they can chat. Your task is to figure out the number of gamers in the longest contiguous group after these line swaps have taken place.

Input The first line of input will be an integer n, $1 \leq n \leq 100$ that indicates the number of people in line. The next line of input will be an integer d, $1 \leq d \leq 10$ that indicates the maximum distance between gamers at which the gamer closest to the front will swap. The final line of input contains a series of n characters that are either X (non-gamer) or G (gamer). The first character represents the person at the front of the line, the second character is the second person in line, and so on.

Output A number g that indicates the number of gamers in the largest contiguous group within the line after all possible swaps have taken place.

Sample Input

```
10
2
GGXXGGXXXG
```

Sample Output

```
4
```

Explanation There are only two people between the second and third gamers in line, so the second gamer will move in front of the third gamer, and the line will look like:

```
GXXGGGXXXG
```

But now there are only two people between the first gamer and the second one, so the first gamer will move in front of the second one and the line will look like:

```
XXGGGGXXXG
```

The fourth gamer is three away from the fifth one, so no more movement takes place.
There are four gamers in the largest contiguous group, so the answer is 4.

Problem C - Hip Translator

According to the national statistics, a teenager sends/receives 100+ text messages a day. Dr. Orooji's teenage children are no exception but the problem is Dr. O (an old-fashioned, face-to-face communicator) has difficulty reading text messages full of abbreviations (short-hands) sent to him by his children. Dr. O needs your help reading these text messages.

Given the list of abbreviations and a paragraph, you are to expand the text (paragraph) so that Dr. O can read it easily.

Input The first input line contains an integer, n where ($1 \leq n \leq 20$), indicating the number of abbreviations. These abbreviations are on the following n input lines, one per line. Each input line starts in column 1 and contains an abbreviation (1-5 characters, consisting of only lowercase letters and/or digits). The abbreviation is followed by exactly one space, and this is followed by the expanded version of the abbreviation (1-50 characters, consisting of only lowercase letters and spaces; assume the expanded version does not start or end with a space and contains no multiple consecutive spaces between words). Assume that all abbreviations are distinct, i.e., no duplicates.

The list of abbreviations is followed by a positive integer, p, indicating the number of input lines containing the paragraph to be expanded. The paragraph is on the following p input lines. Assume these input lines do not exceed column 50, do not start or end with a space, and each line contains at least one word. The paragraph will contain only lowercase letters, digits, and spaces. Assume that there will not be multiple consecutive spaces in the input paragraph.

A word is defined as a consecutive sequence of letters/digits. Assume that a word will be entirely on one input line, i.e., a word is not broken over two or more lines.

Output Each line of the input paragraph must be on one line of output. The input line must be printed in the output exactly the same (spacing). The only exception is that each abbreviation must be replaced by its expanded version, i.e., when an abbreviation is found in the input, its expanded version must be output.

Note that an abbreviation must match a word completely and not just part of a word. For example, if `u` is an abbreviation for `you`, then `u` must appear as a word by itself in the paragraph in order to be replaced, i.e., if the abbreviation is part of a word in the paragraph (e.g., the paragraph contains the word `buy` or `ugly` or `you`), the `u` in these words should not be replaced.

Problem C - Hip Translator (continued)

Sample Input

```
8
g2g got to go
g good
c see
l8 late
l8r later
d i am done
u you
r are
6
hi
how r u
you tell me
you are l8
d
c u l8r
```

Sample Output

```
hi
how are you
you tell me
you are late
i am done
see you later
```

Problem D - Child Check

Dr. Orooji's twins (Mack and Zack) play soccer. We will assume Mack wears jersey number 18 and Zack wears 17. So, Dr. O has to look for these two numbers when trying to find the twins.

Given a list of 10 jersey numbers, determine if the twins are in the group of players on the field.

Input The first input line contains a positive integer, n, indicating the number of data sets to check. The sets are on the following n input lines, one set per line. Each set consists of exactly 10 single-space-separated distinct integers (each integer between 11 and 99 inclusive) giving the jersey numbers for the players.

Output Print each input set. Then, on the next output line, print one of four messages: `mack`, `zack`, `both`, or `none` to indicate how many of the twins are in the set. Leave a blank line after the output for each test case.

Sample Input

```
4
11 99 88 17 19 20 12 13 33 44
11 12 13 14 15 16 66 88 19 20
20 18 55 66 77 88 17 33 44 11
12 23 34 45 56 67 78 89 91 18
```

Sample Output

```
11 99 88 17 19 20 12 13 33 44
zack

11 12 13 14 15 16 66 88 19 20
none

20 18 55 66 77 88 17 33 44 11
both

12 23 34 45 56 67 78 89 91 18
mack
```

Problem E - Cover Up

When going to your internship you got a nice apartment with a skylight. However, one crazy party later and you now have a square-shaped hole where your skylight used to be. Rather than telling the landlord, you decided you would "fix" it by putting a circular pot to collect the water but, as the saying goes, round peg square hole. You need to determine how much of the square the circle covers to help you determine if you should buy a larger pot. Don't worry about the area not covered; you can do multiplication and subtraction easily in your head.

Given the radius of a circular pot and the length of the square skylight, calculate the amount of skylight rain area covered by the pot assuming the two shapes have the same center (i.e., are coaxial) with respect to the direction rain falls from (up). In other words, the center of the square will be directly above the center of the circle.

Input The first input line contains a positive integer, n, indicating the number of test cases. Each of the following n lines contains a pair of integers, s and r where $(1 \leq s, r \leq 100)$, which represent the length of the side of the skylight and the radius of the pot, respectively.

Output For each scenario, output a single decimal representing the area under the skylight that is covered by the pot. Round the answers to two decimal places (e.g., 1.234 rounds to 1.23 and 1.235 rounds to 1.24). For this problem, use 3.14159265358979 as the value of pi.

Sample Input

```
3
1 1
8 5
10 4
```

Sample Output

```
1.00
62.19
50.27
```

Problem F - Blaster Shield

Jedi Knights are often tasked with protection. Whether protecting shield generators or important diplomats, the Jedi will use their lightsabers to deflect blaster shots and keep their asset safe.

Sometimes a Jedi will go on missions alone or will be accompanied by another Jedi. When protecting an asset, they will stand side by side deflecting shots that would otherwise harm the asset they wish to protect. Sometimes, even together, they cannot block all the blaster shots. That is because Jedi are still limited by their physical reaction time. More specifically, when a Jedi blocks a blaster shot, he has to wait a certain amount of time before he can block another shot. For example, a Jedi that takes t time units between shots can block a shot arriving at time k and a shot arriving at time k + t (or later).

So, Jedi will coordinate which shots they each will block and which shots they will allow to pass through their defense. Either Jedi can independently block each shot as long as there is enough time between the current shot and his last blocked shot. But determining which shots to block and which to let by is no easy task if they want to minimize the number of shots that reach their asset. That is why they seek aid from the other great power in the galaxy, programming.

Given the times the blasters reach the Jedi, the number of Jedi, and their reaction time, determine the number of blaster shots that reach the asset they are trying to protect. Note that each Jedi can block a blaster shot at the beginning of the mission but after his first block the Jedi is limited by his reflexes (the time he has to wait before he can block another shot).

Input The first input line contains a positive integer, n, indicating the number of protection missions the Jedi have been assigned. This is followed by the data for each mission.

The first input line for each mission contains an integer, b, $(1 \leq b \leq 1,000)$, the number of blaster shots fired at their asset. This is followed by a line containing b numbers, separated by spaces, which are the times the blaster shots will reach the Jedi. These numbers can be in any order but will be positive integers less than or equal to 1,000. This is followed by an integer j, $(1 \leq j \leq 2)$ on a line by itself, which is the number of Jedi on the mission.

The last input line for a mission contains j space separated integers giving the reaction time of each Jedi, the time it takes a Jedi to prepare to block the next blaster shot. These numbers will be between 1 and 100 inclusive.

Output For each mission, output "Mission #m: a" where m is the mission number (starting with 1) and a is the minimum number of blaster shots that the Jedi are unable to block and will hit their asset. Leave a blank line after the output for each mission.

Problem F - Blaster Sheild (continued)

Sample Input

```
4
5
10 5 5 10 5
1
100
4
2 4 9 9
2
10 7
5
2 4 8 13 13
2
10 7
5
2 4 6 8 10
1
2
```

Sample Output

```
Mission #1: 4
Mission #2: 1
Mission #3: 1
Mission #4: 0
```

Explanation of the Sample Input/Output:

- In Mission #2, Jedi with speed 7 can block 2 and one 9 and Jedi with speed 10 can block 4, resulting in 1 shot remaining unblocked.
- In Mission #3, Jedi with speed 7 can block 4 and one 13 and Jedi with speed 10 can block 2 and the other 13, resulting in 1 shot remaining unblocked.

Problem G - Canoe Crawl

The Phang Nga Bay in Thailand is home to hundreds of islands, some of which (e.g., James Bond Island) are famous from movie scenes. Others have lagoons inside that can only be reached by canoeing through caves on the water. Some caves have ceilings so low that canoers must lean over to make it through.

Besides navigating the tricky passages of the caves, canoers must be aware of the tides. Some caves can only be traversed in low tide. As the tides change, the sea level rises or falls while the explorers are paddling, and they must be careful to choose the correct path through the cave to avoid getting trapped. Of course, if they can make it through, they also want to minimize the amount of energy they spend leaning over in the canoe, i.e., they prefer higher ceiling heights when going through the caves.

In this problem, we assume that canoers start their journey at low tide and the sea level rises by one millimeter each second. Each cave is described by a two-dimensional grid (table) of numbers, where the j^{th} number in the i^{th} row indicates the initial height in millimeters of the cave ceiling at position (i, j) on the surface of the water. Because of the sea level change, the ceiling height (the distance from the sea level to the ceiling) at each cell decreases over time.

The cave can be traversed by starting at the first column of the first row (i.e., northwest corner) and ending at the last column of the last row (i.e., southeast corner). The canoe only moves in one of four directions in the two-dimensional grid (north, south, east, or west), one move at a time. Each second, the canoe moves to an adjacent cell and the sea level increases by one millimeter, and the height of the cell above sea level must be greater than zero when the canoe enters it. The canoe's move and sea level change happen simultaneously, so the ceiling height may become zero just as the canoe is leaving. You may assume that the height of the cave above the canoe's initial position is greater than zero

Given the description of a cave, you must find the path with the highest minimum ceiling height, or determine that it is impossible to traverse. Note that the number of cells the path goes through is not important; rather the heights of the cells are important; in particular, you are to find the path with the largest minimum ceiling height.

Problem G - Canoe Crawl (continued)

Input The first line of input contains a single positive integer, n, indicating the number of caves to process. This is followed by n cave descriptions. Each cave description begins with a line containing two integers, r and c where $1 \leq r \leq 500$ and $1 \leq c \leq 500$, denoting the number of rows and columns, respectively. The next r lines each contain c space-separated integers, with the j^{th} number on the i^{th} line representing the height $a_{i,j}$ where $(0 \leq a_{i,j} \leq 10^9; a_{1,1} > 0)$ in millimeters of the cave ceiling above the initial sea level.

Output For each cave, output a line with a single integer k denoting the largest minimum ceiling height, in millimeters, of a path through the cave, or the word "impossible" if the cave can't be traversed.

Sample Input

```
2
4 5
9 5 4 0 0
9 4 8 9 12
9 6 8 7 12
0 0 9 8 12
3 1
10
1
10
```

Sample Output

```
3
impossible
```

Problem H - Currency Calculator

Frank Marks works at the Business Office of a large company. His company has customers all over the world and must deal with many different currencies. Employees often come to the Business Office with requisitions for certain amounts of money, such as 100 American dollars or 452 Euros. If Frank has the cash on hand, he gives the employee exactly what they need; if he does not have enough of the particular currency requested, he substitutes it with another one. This is sometimes difficult because he may have many different currencies to choose from and would like to pick the one which allows him to get as close to the original requisition as possible without going under (he must provide at least the value requested). For example, suppose Frank has six difference currencies – A, B, C, D, E and F – and he is aware of the following exchange rates:

$$23\ A = 17\ B$$

$$16\ C = 29\ E$$

$$5\ B = 14\ E$$

$$1\ D = 7\ F$$

If a requisition for 100 A comes in but Frank has less than 100 A available, he can substitute with either 74 B (equivalent to about 100.12 A), 115 C (equivalent to about 100.72 A) or 207 E (equivalent to about 100.02 A). Thus, the best approximation available to him is 207 E. Note that Frank does not have enough information to find a relationship between currencies A and D or currencies A and F. Also, Frank only has at most 100,000 units of any one currency in stock, so he could not satisfy a request for 64,000 A using E currency and would need to use 73,078 C instead. Determining the ideal substitute currency to use when he has completely run out of the requested currency is time consuming, so Frank would like a program to do the calculations for him.

Problem H - Currency Calculator (continued)

Input Each test case begins with a positive integer n indicating the number of exchange rates. The next n lines will be of the form

$$val_1\ name_1 = val_2\ name_2$$

where $name_1$ and $name_2$ are the names of two distinct currencies, and val_1 and val_2 are positive integers ≤ 30 specifying the ratio between the two currencies. There will be no more than 8 distinct currency names, and any two currencies will be paired together at most once. Currency names will consist of up to ten alphabetic characters. There will be no inconsistencies in the input (such as $1A = 2B$, $1B = 2C$ and $1C = 2A$). Following these n lines will be a single line of the form

$$val\ name$$

which specifies the amount (a positive integer not exceeding 100,000) and the name of the currency requested. A line containing 0 will follow the last test case.

Output For each test case, print the case number and the closest approximation without going under the requested value assuming that Frank does not have any of the requested currency available but is fully in stock of all other currencies. There will be a unique answer for each problem instance.

Sample Input

```
4
23 A = 17 B
16 C = 29 E
5 B = 14 E
1 D = 7 F
100 A
1
1 shekel = 2 quatloo
40 quatloo
0
```

Sample Output

```
Case 1: 207 E
Case 2: 20 shekel
```

Problem I - Stacking Plates

The *Plate Shipping Company* is an Internet retailer that, as their name suggests, exclusively sells plates. They pride themselves in offering the widest selection of dinner plates in the universe from a large number of manufacturers.

In a recent cost analysis the company has discovered that they spend a large amount of money on packing the plates for shipment. Part of the reason is that plates have to be stacked before being put into shipping containers. And apparently, this is taking more time than expected. Maybe you can help.

A shipment of plates consists of plates from several manufacturers. The plates from each manufacturer come stacked, that is, each arranged in a single stack with plates ordered by size (the smallest at the top, the largest at the bottom). We will call such a stack properly ordered. To ship all these plates, you must combine them into a single stack, again properly ordered. To join the manufacturers' stacks into a single stack, two kinds of operations are allowed:

- Split: a single stack can be split into two stacks by lifting any top portion of the stack and putting it aside to form a new stack.
- Join: two stacks can be joined by putting one on top of the other. This is allowed only if the bottom plate of the top stack is no larger than the top plate of the bottom stack, that is, the joined stack has to be properly ordered.

Note that a portion of any stack may never be put directly on top of another stack. It must first be split and then the split portion must be joined with the other stack. Given a collection of stacks, you have to find the minimum number of operations that transforms them into a single stack. The following example corresponds to the sample input, and shows how two stacks can be transformed to a single stack in five operations:

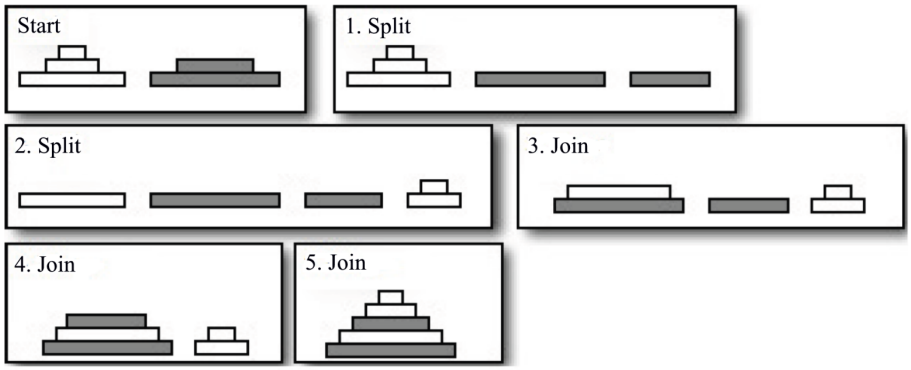

Figure 1: Figure I.1: Image of series of swaps and joins

Problem I - Stacking Plates (continued)

Input Each test case starts with a line containing a single integer n, ($1 \leq n \leq 50$), the number of stacks that have to be combined for a shipment. This is followed by n lines, each describing a stack. These lines start with an integer h, ($1 \leq h \leq 50$), the height of the stack. This number is followed by h positive integers that give the diameters of the plates, from top to bottom. All diameters are at most 10,000. These numbers will be in non-decreasing order.

Output For each test case, display the case number and the minimum number of operations (splits and joins) that have to be performed to combine the given stacks into a single stack.

Sample Input

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

Sample Output

```
Case 1: 5
Case 2: 2
```

Problem J - Short Crypt

You've been tasked with relaying coded messages to your fellow resistance fighters. Each coded message is a sequence of lower-case letters that you furtively scrawl on monuments in the dead of night.

Since you're writing these messages by hand, the longer the message, the greater the likelihood of being caught by the evil empire while writing. Because of this you decide it would be worthwhile to come up with a simple encoding that might allow for shorter messages. After thinking about it for a while, you decide to use integers and parentheses to indicate repetition of substrings when doing so shortens the number of characters you need to write. For example, the 10 character string

$$abcbcbcbca$$

could be more briefly written as the 7 character string

$$a4(bc)a$$

If a single letter is being repeated, parentheses are not needed. Also, repetitions may themselves be repeated, so you can write the 20 character string

$$abbbcdcdcdabbbcdcdcd$$

as the 11 character string

$$2(a3b3(cd))$$

Input Each test case consists of a single line containing a string of lowercase letters of length l where $0 < l \leq 500$. A line containing a single 0 will terminate the input.

Output For each test case, output the number of characters needed for a minimal encoding of the string.

Problem J - Short Crypt (continued)

Sample Input

```
abcbcbcbca
abbbcdcdcdabbbcdcdcd
0
```

Sample Output

```
Case 1: 7
Case 2: 11
```

Problem K - Comma Sprinkler

As practice will tell you, the English rules for comma placement are complex, frustrating, and often ambiguous. Many people, even the English, will, in practice, ignore them, and, apply custom rules, or, no rules, at all.

Doctor Comma Sprinkler solved this issue by developing a set of rules that sprinkles commas in a sentence with no ambiguity and little simplicity. In this problem you will help Dr. Sprinkler by producing an algorithm to automatically apply her rules.

Dr. Sprinkler's rules for adding commas to an existing piece of text are as follows:

1. If a word anywhere in the text is preceded by a comma, find all occurrences of that word in the text, and put a comma before each of those occurrences, except in the case where such an occurrence is the first word of a sentence or already preceded by a comma.
2. If a word anywhere in the text is succeeded by a comma, find all occurrences of that word in the text, and put a comma after each of those occurrences, except in the case where such an occurrence is the last word of a sentence or already succeeded by a comma.
3. Apply rules 1 and 2 repeatedly until no new commas can be added using either of them.

As an example, consider the text:

```
please sit spot. sit spot, sit. spot here now here.
```

Because there is a comma after `spot` in the second sentence, a comma should be added after `spot` in the third sentence as well (but not the first sentence, since it is the last word of that sentence). Also, because there is a comma before the word `sit` in the second sentence, one should be added before that word in the first sentence (but no comma is added before the word `sit` beginning the second sentence because it is the first word of that sentence). Finally, notice that once a comma is added after `spot` in the third sentence, there exists a comma before the first occurrence of the word `here`. Therefore, a comma is also added before the other occurrence of the word `here`. There are no more commas to be added so the final result is:

```
please, sit spot. sit spot, sit. spot, here now, here.
```

Problem K - Comma Sprinkler (continued)

Input The input contains one line of text, containing at least 2 characters and at most 1,000,000 characters.

Each character is either a lowercase letter, a comma, a period, or a space. We define a word to be a maximal sequence of letters within the text. The text adheres to the following constraints:

- The text begins with a word.
- Between every two words in the text, there is either a single space, a comma followed by a space, or a period followed by a space (denoting the end of a sentence and the beginning of a new one).
- The last word of the text is followed by a period with no trailing space.

Output Display the result after applying Dr. Sprinkler's algorithm to the original text.

Sample Input 1

```
please sit spot. sit spot, sit. spot here now here.
```

Sample Output 1

```
please, sit spot. sit spot, sit. spot, here now, here.
```

Sample Input 2

```
one, two. one tree. four tree. four four. five four. six five.
```

Sample Output 2

```
one, two. one, tree. four, tree. four, four. five, four. six five.
```

Problem L - Amalgamated Artichokes

Fatima Cynara is an analyst at Amalgamated Artichokes (AA). As with any company, AA has had some very good times as well as some bad ones. Fatima does trending analysis of the stock prices for AA, and she wants to determine the largest decline in stock prices over various time spans. For example, if over a span of time the stock prices were 19, 12, 13, 11, 20 and 14, then the largest decline would be 8 between the first and fourth price. If the last price had been 10 instead of 14, then the largest decline would have been 10 between the last two prices.

Fatima has done some previous analyses and has found that the stock price over any period of time can be modelled reasonably accurately with this equation:

$$price(k) = p \times (sin(a \times k + b) + cos(c \times k + d) + 2)$$

where p, a, b, c and d are constants. Fatima would like you to write a program to determine the largest price decline over a given sequence of prices. Figure C.1 illustrates the price function for Sample Input 1. You have to consider the prices only for integer values of k.

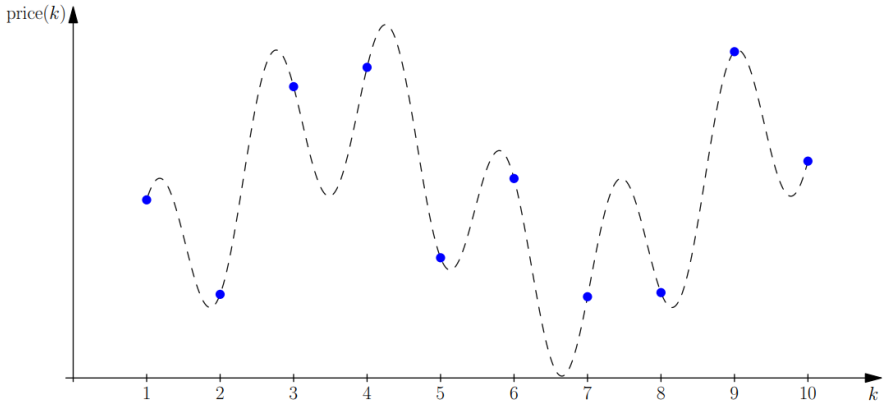

Figure 2: Illustration of price function

Figure L.1: Sample Input 1. The largest decline occurs from the fourth to the seventh price.

Problem L - Amalgamated Artichokes (continued)

Input The input consists of a single line containing 6 integers p, a, b, c, d, n where:

- $1 \leq p \leq 1000$
- $0 \leq a, b, c, d \leq 1000$
- $1 \leq n \leq 106$

The sequence of stock prices to consider is $price(1), price(2), \ldots, price(n)$.

Output Display the maximum decline in the stock prices. If there is no decline, display the number 0.00. Your output should have an absolute or relative error of at most 10^{-6}.

Sample Input 1

```
42 1 23 4 8 10
```

Sample Output 1

```
104.855110477
```

Sample Input 2

```
100 7 615 998 801 3
```

Sample Output 2

```
0.00
```

Sample Input 3

```
100 432 406 867 60 1000
```

Sample Output 3

```
399.303813
```

Problem M - Treasure

Who doesn't love a good treasure hunt?

The devious pirate BashBeard has buried his treasure in a 5x5 ARRRRRRRRay (that what pirates call an array). Fortunately he left you a map.

The first cell of the array (row 1, column 1) contains a clue. This clue will be a two digit number referring to the row and column of the next clue. Follow the clues until you find a clue that references itself, there you will find the treasure.

All clues will be two digit numbers in *row column* format, where all rows and columns will be integers i, $1 \leq i \leq 5$. A clue of 35 would indicate the next clue is in row 3 column 5. Access that cell and look for another clue. Continue following the clues until the cell you visit has a value that is equal to its location in the array.

A sample BashBeard map would look like:

	column 1	column 2	column 3	column 4	column 5
row 1	55	21	32	41	25
row 2	21	42	43	14	31
row 3	54	45	52	42	23
row 4	33	15	51	31	35
row 5	21	52	33	13	51

Starting at 11 (top left corner) we read the clue 55, reading the clue at 55 takes us to 51, then to 21.
The clue in cell 21 is 21 thus we have found the treasure and our output should be 21.

Input Input will consist of 25 two-digit numbers, one per line. They correspond to the clues that belong in the following locations:

```
row1column1
row1column2
row1column3
row1column4
row1column5
row2column1
...
```

and so on. Construct the 5x5 array, begin at 11, and find the treasure.

Output Output will consist of the location of the treasure in two-digit clue form.

Problem M - Treasure (continued)

Sample Input For the example in the problem statement the input would be:

```
55
21
32
41
25
21
42
43
14
31
54
45
52
42
23
33
15
51
31
35
21
52
33
13
51
```

Sample Output The expected output would simply be:

```
21
```