CODEVITA QUESTIONS - ZONE:1

Codekart

Problem Description

Codu wants to create a shopping application. The application would sell only SHIRT and SHOE and have a cost that can be modified based on market needs. This application should allow users in two roles, viz. store manager(SM) and shopper(S).

Codu wants to test the app. He wants the application to execute a few commands and print the output.

Following is the list of allowed **commands**:

CMD SM ADD [ITEM NAME] [ITEM QTY] - adds the given quantity of the item to the inventory and prints(returns) the quantity added, otherwise prints -1 when there is any error or invalid input. Item qty can only be whole numbers > 0.

CMD SM REMOVE [ITEM NAME] - removes the item from the inventory, returns and prints -1 when there is an error, otherwise prints(returns) 1.

CMD SM GET_QTY [ITEM NAME] - returns and prints the currently available quantity for the item in the inventory, otherwise prints(returns) 0 in case the item is not found.

CMD SM INCR [ITEM NAME] [ITEM QTY] - adds the given quantity of the item to the inventory and prints(returns) the quantity added, otherwise prints(returns) -1 when there is any error or invalid input. item qty can only be whole numbers > 0.

CMD SM DCR [ITEM NAME] [ITEM QTY] - removes the given quantity of the item from the inventory and prints(returns) the quantity added, otherwise prints(returns) -1 when there is any error or invalid input. item qty can only be whole numbers > 0.

CMD SM SET_COST [ITEM NAME] [COST] - sets the cost of the item, returns the value, otherwise prints -1 in case of any errors or invalid input. Cost must be decimal.

CMD S ADD [ITEM NAME] [ITEM QTY] - adds the given quantity of the item to the shopping cart and prints(returns) the quantity added, otherwise prints -1 when there is any error or invalid input. Item qty can only be whole numbers > 0.

CMD S REMOVE [ITEM NAME] - removes the item from the shopping cart, returns and prints -1 when there is an error, otherwise prints(returns) 1.

CMD S INCR [ITEM NAME] [ITEM QTY] - adds the given quantity of the item to the shopping cart and prints(returns) the quantity added, otherwise prints(returns) -1 when there is any error or invalid input. item qty can only be whole numbers > 0.

CMD S DCR [ITEM NAME] [ITEM QTY] - removes the given quantity of the item from the shopping cart and prints(returns) the quantity added, otherwise prints(returns) -1 when there is any error or invalid input. item qty can only be whole numbers > 0.

CMD S GET_ORDER_AMOUNT - gets the total price of the items in the cart, returns the value, otherwise prints -1 in case of any error or invalid input. The total amount should be rounded and printed up to two decimal places.

NOTE- Increment and Decrement operations are only possible when the item is already in the inventory or cart. If increment or decrement is attempted on items that do not exist in the cart, then the command should return and print -1.

If an attempt is made to add an item that is already in the inventory or cart, such operations should result in an error and must return and print -1.

If an item which is present in the cart of inventory is removed using *remove* command and an increment or decrement operation is performed on it, such operations should result in an error and must return and print -1.

If any item quantity after decrement becomes zero, the same is removed from the corresponding inventory or cart. Performing increment or decrement operation after such a previous decrement operation, should result in an error and return -1.

You need to think of other similar error conditions while implementing the solution.

Please note at the beginning of a test case or command set, both the inventory as well as the cart is empty.

Here,

SM= STORE MANAGER

S= SHOPPER

You are required to create the application for Codu to manage the shopping kiosk.

The first line of input **T**, gives the number of test cases.

Each test set is a set of commands, which ends with "END" string.

Each command in a test case is on a new line

Constraints

1<=T<=10

Input

First line contains an integer T, which denotes the number of test cases

Second line onwards, there will be commands until we receive END command. Any command after the END command belongs to next test case.

For command format refer to Example section.

Output

Print output of every command. (Print double value for command SET_COST(rounding to one decimal places) and GET_ORDER_AMOUNT(rounding to two decimal places))

Time Limit

1

Examples

Example 1

Input

```
1

CMD SM SET_COST SHOE 5

CMD SM SET_COST SHIRT 10

CMD SM ADD SHOE 5

CMD SM ADD SHIRT 10

CMD SM DCR SHIRT 5

CMD SM INCR SHOE 5

CMD SM GET_QTY SHIRT

CMD SM GET_QTY SHOE

CMD SM REMOVE SHIRT

CMD SM GET_QTY SHIRT

CMD S ADD SHOE 2

CMD S INCR SHOE 2

CMD S DCR SHOE 1
CMD S GET_ORDER_AMOUNT

END
```

Output

```
5.0

10.0

5

10

5

5

5

10

1

0

2

2

1

15.00
```

Explanation :

From commands "**CMD SM SET_COST SHOE 5**" and "**CMD SM SET_COST SHIRT 10**"

We are successfully setting the cost as 5.0 and 10.0 respectively.

From next commands "**CMD SM ADD SHOE 5**" and "**CMD SM ADD SHIRT 10"**

Quantity of 5 shoes and 10 shirts has been successfully added to the inventory.

From next commands "**CMD SM DCR SHIRT 5**"" and "**CMD SM INCR SHOE 5"**

Shirt quantity is decremented by 5 and shoe quantity is incremented by 5. This leaves us with 5 shirts and 10 shoes in the inventory.

From next commands "**CMD SM GET_QTY SHIRT**" and "**CMD SM GET_QTY SHOE"**

We are getting the quantity of shirt and shoe, which is 5 and 10 respectively.

From next command "**CMD SM REMOVE SHIRT"**

Shirt is removed from the inventory and hence 1 is printed.

From next command "**CMD SM GET_QTY SHIRT"**

We are querying the quantity of shirt, which is 0 as it was removed in the previous command.

From next command "**CMD S ADD SHOE 2"**

Shopper adds two shoes to the cart hence 2 is printed.

From next commands "**CMD S INCR SHOE 2**" and "**CMD S DCR SHOE 1**"

The user increments these shoes by 2 and then decrements by 1 hence 2 and 1 are printed. SO current shoes in cart= 2+2-1=3

From next commands "**CMD S GET_ORDER_AMOUNT**"

The next command asks to print order amount or cart value, which is the cost of shoes * the number of shoes=5*3=15 hence 15.00 is printed by rounding to two decimal places.

Prime Time Again

Problem Description

Here on earth, our 24-hour day is composed of two parts, each of 12 hours. Each hour in each part has a corresponding hour in the other part separated by 12 hours: the hour essentially measures the duration since the start of the day part. For example, 1 hour in the first part of the day is equivalent to 13, which is 1 hour into the second part of the day.

Now, consider the equivalent hours that are both prime numbers. We have 3 such instances for a 24-hour 2-part day:

5~17

7~19

11~23

Accept two natural numbers D, P >1 corresponding respectively to number of hours per day and number of parts in a day separated by a space. D should be divisible by P, meaning that the number of hours per part (D/P) should be a natural number. Calculate the number of instances of equivalent prime hours. Output zero if there is no such instance. Note that we require each equivalent hour in each part in a day to be a prime number.

Example:

Input: 24 2

Output: 3 (We have 3 instances of equivalent prime hours: 5~17, 7~19 and 11~23.)

Constraints

10 <= D < 500

2 <= P < 50

Input

Single line consists of two space separated integers, D and P corresponding to number of hours per day and number of parts in a day respectively

Output

Output must be a single number, corresponding to the number of instances of equivalent prime number, as described above

Time Limit

1

Examples

Example 1

Input

36 3

Output

2

Explanation

In the given test case D = 36 and P = 3

Duration of each day part = 12

2~14~X

3~15~X

5~17~29 - instance of equivalent prime hours

7~19~31 - instance of equivalent prime hours

11~23~X

Hence the answers is 2.

Example 2

Input

49 7

Output

0

Explanation

Duration of each day part = 7

2~9~X~23~X~37~X

3~X~17~X~31~X~X

5~X~19~X~X~X~47

7~X~X~X~X~X~X

Hence there are no equivalent prime hours.

Number Distancing

Problem Description

Consider 9 natural numbers arranged in a 3x3 matrix:

n11 n12 n13

n21 n22 n23

n31 n32 n33

Define numbers "in contact" with a given number to be those that appear closest to it on the same row, column or diagonally across:

Contacts of number n11: n12, n22 and n21

Contacts of number n12: n11, n21, n22, n23, n13

Contacts of number n13: n12, n22, n23

Contacts of number n21: n11, n12, n22, n32, n31

Contacts of number n22: n11, n12, n13, n23, n33, n32, n31, n21

Contacts of number n23: n13, n12, n22, n32, n33

Contacts of number n31: n21, n22, n32

Contacts of number n32: n31, n21, n22, n23, n33

Contacts of number n33: n32, n22, n23

The problem now is that numbers having a common factor (other than 1) should not be "in contact". In other words, a pair of numbers can remain neighbours only if their highest common factor is 1.

The following rules apply to enforce this "distancing":

1. The central number (n22) stays put.

2. The corner numbers (n11, n13, n33, n31) can move in the same row or column or diagonally away from the centre.

3. The numbers "on the walls" (n12, n23, n32, n21) can only move from the walls i.e. n21 can only move "left", n12 can only move "up", n23 can only move "right" and n32 can only move "down".

4. Each number should stay put as far as possible and the "distancing" operation should result in the least number of numbers ending up without any contacts.

5. After satisfying rule 4, if there are multiple options for the final matrix, then the "distancing" operation should result in the smallest (m x n matrix, including the intervening blank space elements, with the least possible value of m*n).

6. If, after satisfying all the rules above, there are multiple distancing options for a set of numbers, the largest number keeps to its original cell.

Constraints

1 <= Element of grid <= 100

Input

First line consists of 9 space separated integers denoting n11, n12, n13, ... n23, n33 respectively.

Output

Print the "contact" less numbers in ascending order of their value separated by space. Output "None" if there are no such numbers.

Time Limit

1

Examples

Example 1

Input

23 33 12 1 2 5 25 6 10

Output

10

Explanation

Initial configuration

23 33 12

1 2 5

25 6 10

The optimal distancing options result in the following possibility (space denoted by *):

23 33 * 12

1 2 5 *

25 * * *

* 6 * 10

10 ends up as the number without contacts.

Example 2

Input

1 2 3 4 5 6 7 8 9

Output

None

Explanation

Initial configuration:

1 2 3

4 5 6

7 8 9

The optimal distancing options result in the following 5x3 matrix (space denoted by *):

* 2 3

1 * *

4 5 6

7 * *

* 8 9

There is finally no number without a contact.

Example 3

Input

2 6 2 10 19 12 2 20 2

Output

2 2 2 2 10 12

Explanation

Initial Configuration:

Approach 1) Moving 6 and 20

Final Matrix

2 * 6 * 2

* * * * *

* 10 19 12 *

* * * * *

2 * 20 * 2

Approach 2) Moving 10 and 12

2 * * * 2

* * 6 * *

10 * 19 * 12

* * 20 * *

2 * * * 2

We prefer approach 2) and not 1) since the largest of all the elements (20) needs to be retained in it's original cell.

Faulty Keyboard

Problem Description

Mr. Wick has a faulty keyboard. Some of the keys of the keyboard don't work. So he has copied all those characters corresponding to the faulty keys on a clipboard. Whenever those characters need to be typed he pastes it from the clipboard. In typing whatever is required he needs to make use of paste, backspace and cursor traversal operations. Help him in minimize the number of operations he needs to do to complete his typing assignment. Each operation has one unit weightage.

Constraints

1 <= S <= 16

1 <= T<= 10^4

String T and S will only be comprised of letters a-z and digits 0-9

Input

First line contains text T to be typed Second line contains string S of all the faulty keys pasted on clipboard

Output

Print the minimum number of operations required for typing the text T

Time Limit

1

Examples

Example 1

Input

experience was ultimate

ew

Output

14

Explanation

experience =(2+2+2+2) =[{p+b} + {p+b} +{p+b} +{p+b}]

was=(4)=[p+m+b+m]

ultimate=(2)=[p+b]

where p=paste, b=backspace ,m= move cursor

Example 2

Input

supreme court is the highest judicial court

su

Output

17

Explanation

supreme =(1) =[p]

court=(4)=[p+m+b+m]

is=(2)=[p+b]

the=(0)

highest=(2)=[p+b]

judicial=(4)=[p+m+b+m]

court=(4)=[p+m+b+m]

Travel Cost

Problem Description

Suresh wants to travel from city A to city B. But due to coronavirus pandemic, almost all the cities have levied entry tax into the cities so that the number of people entering the city can be limited. Suresh can skip at the most m cities at a time. Suresh has to declare his itinerary at the time of leaving city A. Thus he will have to pay upfront for entire itinerary and also has to pay a fee to get the slips issued. Upon payment he will be given the slips for intermediate cities where he has to show the slips to pass through, en route to his destination.

Some cities have enforced lockdown, that means those cities have blocked the entry into the cities and you will have to skip the cities in any case, such cities are represented by -1. This information is known to Suresh upfront.

Help Suresh find the minimum amount to be paid to reach from City A to City B

Constraints

No of cities <=10^5

Input

First line contains an integer N, denoting the number of cities

Second line contains N space separated integers, where first integer denotes the cost of issuing itinerary slips and next (N-1) integers denote the entry fee of all cities. The last integer is always the destination city. If city is under lock down then its entry fee will be -1.

Third line contains an integer, M which represents number of cities he can skip from a present city during his travel

Output

Single integer which represents the minimum cost Suresh has to pay to travel from city A to B, but if city B is not reachable then print -1

Time Limit

1

Examples

Example 1

Input

5

1 6 -1 5 7

1

Output

19

Explanation

Since he could skip only 1 city between the cities. He will have to pay 1+6+5+7, where 1 is the fees paid to issue slips and [6,5,7] are the fees paid for the entry to the respective cities. So the total amount he has to pay while leaving A is 19.

Example 2

Input

4

3 4 1 -1

3

Output

-1

Explanation

Since the city B is under lockdown, he cannot go to city B. Hence the output is -1

SudoKube

Problem Description

John, a research scholar / Professor / Puzzle solver wants your help in publishing his work on SudoKube on his online blog for his followers and students.

A SudoKube is a mixture of Rubics cube and Sudoku. A SudoKube has exactly 6 appearances of every digit from 1 to 9 across the cube, whereas Rubics cube has 6 different colours.

As John wants to publish his work in text /document form (no video) he's concerned how he would depict the step by step work of rotation in 2D form. Following are the notions and concepts John follows:

1. The six faces of the cube are named FRONT, BACK, UP, DOWN, LEFT and RIGHT respectively.

2. Just like a Rubics cube which move in 90 and 180 degrees in both clockwise and anti clockwise directions, so can the SudoKube

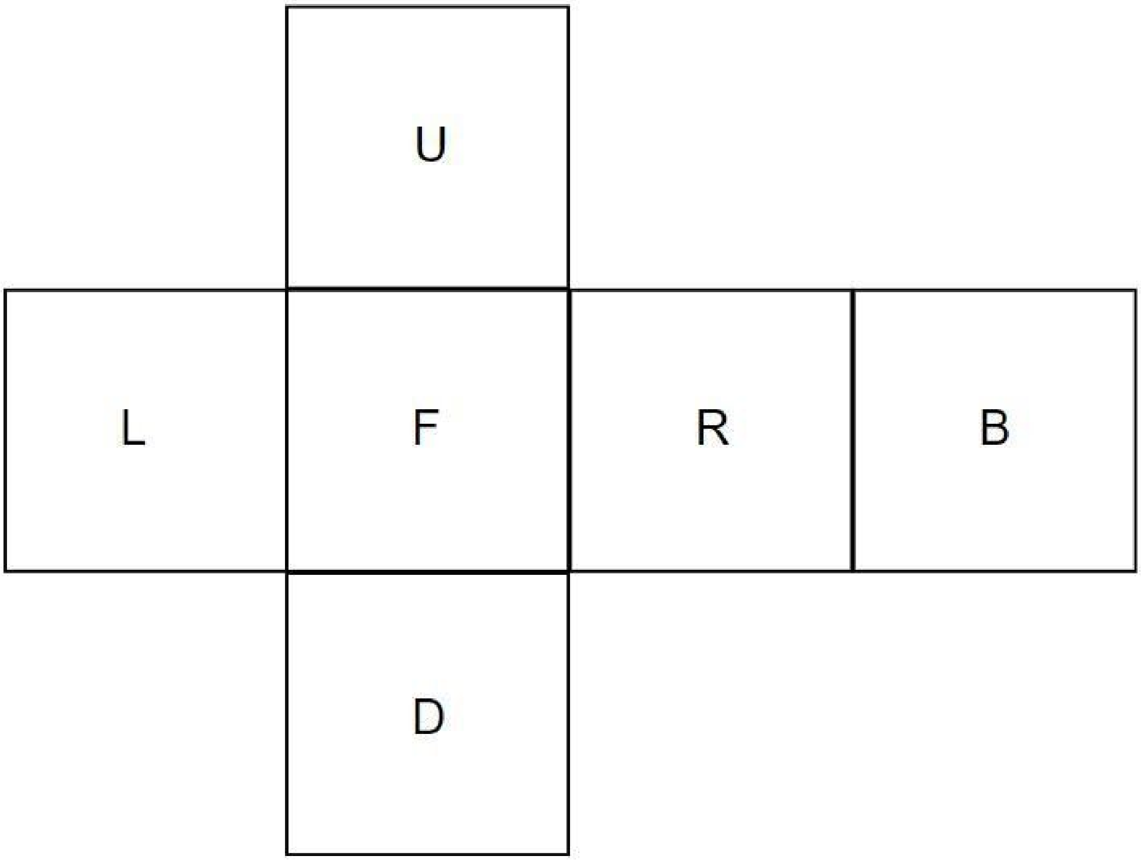
3. Any given face of the cube is a 3x3 square matrix whose indices are denoted by (0,0) to (2,2). Diagram below illustrates the same.

4. An elementary move is denoted in the following fashion.

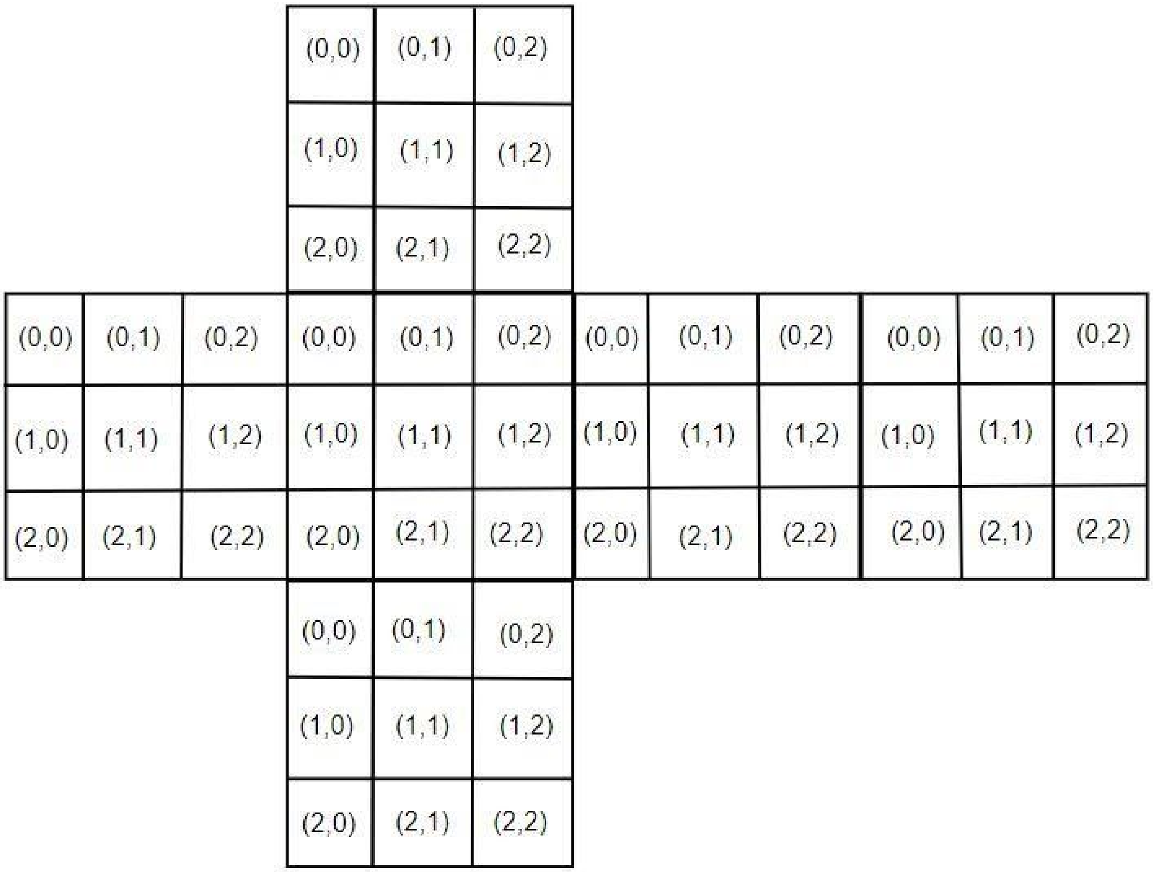
   i. If a given face is rotated by 90 degrees clockwise about the axis passing from the centre of the face to the centre of the cube, the move is denoted by the first letter of the name of the face.

   ii. If the rotation is anticlockwise by 90 degrees, the letter is followed by an apostrophe (').

   iii. If the rotation is by 180 degrees, the letter is followed by a 2.



Above image display the position of the faces



Above diagram displays the indices of the matrix on the faces

John wants to test his notations on you. He has given you the initial position of the SudoKube and he has given you a set of operations to be performed on the SudoKube basis his notation. After applying all the operations, the final SudoKube state should be the same as what John expects. Your task is to apply the operations and print the final SudoKube state.

Constraints

Values in SudoKube will be between 1 and 9

No of moves < 15

Input

- First eighteen lines contain the values of the faces on SudoKube in the order given below

D D D

D D D

D D D

U U U

U U U

U U U

L L L

L L L

L L L

F F F

F F F

F F F

R R R

R R R

R R R

B B B

B B B

B B B

where

- D for Down face
- U for upper face
- L for Left face
- F for Front face
- R for Right face
- B for Back face.

Input contains digits from 1 to 9 instead of letters; letters are displayed for better understanding of the faces and the expected input format

• Nineteenth line contains a sequence of space delimited moves that need to be performed on the SudoKube

Example 1: D F2 R' U - to understand this please refer second example from the *Examples* section below

Example 2: L2 U B F' D2 R - lets understand how to interpret this set of operations

- L2 means rotate the Left side by 180 degrees
- U means rotate the Up side by 90 degrees clockwise
- B means rotate the Back side by 90 degrees clockwise
- F' means rotate the Front side by 90 degrees anticlockwise
- D2 means rotate the Down side by 180 degrees
- R means rotate the Right side by 90 degrees clockwise

In summary, first eighteen lines denotes the state of the SudoKube, 19th line denotes the operation to be performed on that state and output should be the resulting state.

Output

Print 3x3 matrix corresponding to the order (D, U, L, F, R, B). Between every 3x3 matrix there should be a new line.

Time Limit

1

Examples

Example 1

Input

4 7 1

2 8 7

6 3 5

5 8 3

3 1 6

9 4 2

5 2 4

3 7 8

5 1 9

```
6 1 4
9 4 8
2 5 7
7 9 1
1 9 6
6 2 8
8 6 3
7 2 5
3 9 4
F
```

Output

```
6 1 7
2 8 7
6 3 5

5 8 3
3 1 6
9 8 4

5 2 4
3 7 7
5 1 1

2 9 6
5 4 1
```

```
7 8 4

9 9 1
4 9 6
2 2 8

8 6 3
7 2 5
3 9 4
```

Explanation:

The output shows the state of SudoKube when the front side is rotated clockwise

Example 2

Input

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

7 9 1

1 9 6

6 2 8

8 6 3

7 2 5

3 9 4

D F2 R' U

Output

2 4 5

3 8 9

5 7 6

4 3 5

2 1 8

8 7 6

9 1 3

3 7 1

3 9 7

1 6 7

8 4 6

4 1 6

1 6 3

```
9 9 5

4 8 4


5 2 2

7 2 5

9 2 8
```

Explanation

The above output prints the state of cube after D F2 R' U operation are performed. Here

- D means rotate the Down side by 90 degrees clockwise

- F2 means rotate the Front side by 180 degrees

- R' means rotate the Right side by 90 degrees anti clockwise

- U means rotate the Up side by 90 degrees clockwise