Diet Plan

Problem Description

Arnold is planning to follow a diet suggested by his Nutritionist. The Nutritionist prescribed him the total protein, carbohydrates and fats, he should take daily. Arnold searches on an online food store and makes a list of protein, carbohydrates and fats contained in a single unit of various food items.

His target is to have the maximum protein, carbohydrates and fats in his diet without exceeding the prescribed limit. He also wants to have as much diverse food items as much as possible. That is, he does not want to have many units of one food item and 0 of others. Multiple combinations of 'units of food items' are possible to achieve the target. Mathematically speaking, diversity is more if the difference between the number of units of food item chosen the most and the number of units of another food item chosen the least, is as small as possible.

To solve this problem, he uses maximum possible number of units of all the items so that total amount of protein, carbohydrates and fats is not more than prescribed limit. For example - if total nutrition required is 100P, 130C and 130F (where P is Protein, C is Carbohydrates and F is Fats) and 2 different food items, viz. Item A and Item B, have following amount of nutrition:

Item A - 10P, 20C, 30F

Item B - 20P, 30C, 20F

then, he can have (maximum possible) 2 units of all the items as having 3 units will exceed the prescribed amount of Carbohydrates and fats.

Next, he chooses food items to fulfill the remaining nutrients. He chooses one more units of maximum number of food items. He continues this process till he cannot add a unit of any food item to his diet without exceeding the prescribed limit. In this example, he can choose one more unit of item B or one more unit of item A. In case he has two sets of food items then the priority is given to fulfill the requirements of Protein, Carbohydrates, and Fats in that order. So he chooses item B.

You will be provided the maximum nutrients required and the nutrients available in various food items. You need to find the amount of nutrients for which there is a shortfall as compared to the prescription, after making his selection using the process described above. In the example he still needs 20P, 0C, 10F to achieve his target.

Constraints

Number of Food Items <= 10

Maximum amount of Nutrients is less than 1500 i.e. x +y + z <= 1500

Amount of P, C, F in two food items will not be same

P, C, F values in input can be in any order

Output should be in order - P, C, F.

Input

First line contains the maximum limit of nutrients in the following format.

xP yC zF, where x, y and z are integers

Second line contains nutrient composition of different food items separated by pipe (|).

Output

Print the shortfall for each nutrient type, fulfilled separated by space.

E.g. If the output is 10P, 20 C, 30 F, then print "10 20 30" (without quotes).

Time Limit

1

Examples

Example 1

Input

100P 130C 130F

10P 20C 30F|20P 30C 20F

Output

20 0 10

Explanation

Having 2 units of item A and 3 units of item B provides - 2 \* [10P 20C 30F] + 3 \* [20P 30C 20F] = 100P, 130C, 120F. This is the best combination that can reduce the shortfall [20P, 0C, 10F] without exceeding his prescription. In contrast, if 3 units of A and 2 units of B are chosen then 3 \* [10P 20C 30F] + 2 \* [20P 30C 20F] = 70P, 120C, 130F produces a shortfall of [30P, 10C, 0F]. However, since protein shortfall in this case is more than the previous combination, this is not the right combination to follow.

Example 2

Input

130P 120C 110F

4P 9C 2F|4P 3C 2F|7P 1C 3F

Output

2 4 50

Explanation

Having 9 units of item A, 9 units of item B and 8 units of Item C provides - 9 \* [4P 9C 2F] + 9 \* [4P 3C 2F] + 8 \* [7P 1C 3F] = 108P, 116C, 60F. This is the best combination that can reduce the shortfall [2P, 4C, 50F] without exceeding his  prescription

## 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.

## Minimize The Sum

### Problem Description

Given an array of integers, perform atmost K operations so that the sum of elements of final array is minimum. An operation is defined as follows -

Consider any 1 element from the array, arr[i].

Replace arr[i] by floor(arr[i]/2).

Perform next operations on updated array.

The task is to minimize the sum after atmost K operations.

### Constraints

1 <= N, K <= 10^5.

### Input

First line contains two integers N and K representing size of array and maximum numbers of operations that can be performed on the array respectively.

Second line contains N space separated integers denoting the elements of the array, arr.

### Output

Print a single integer denoting the minimum sum of the final array.

### Time Limit

1

### Examples

Example 1

Input

4 3

20 7 5 4

Output

17

Explanation

Operation 1 -> Select 20. Replace it by 10.

New array = [10, 7, 5, 4]

Operation 2 -> Select 10. Replace it by 5.

New array = [5, 7, 5, 4].

Operation 3 -> Select 7. Replace it by 3.

New array = [5,3,5,4].

Sum = 17.

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]

## 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

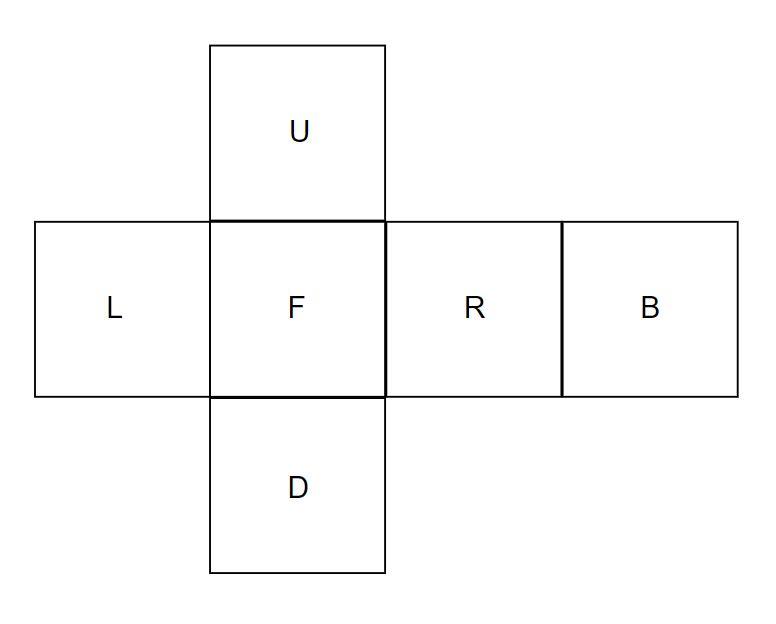
3. Any given face of the cube is a 3x3 square matrix whose indices are denoted by (0,0) to (2,2). Diagram below illustrates the same.

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

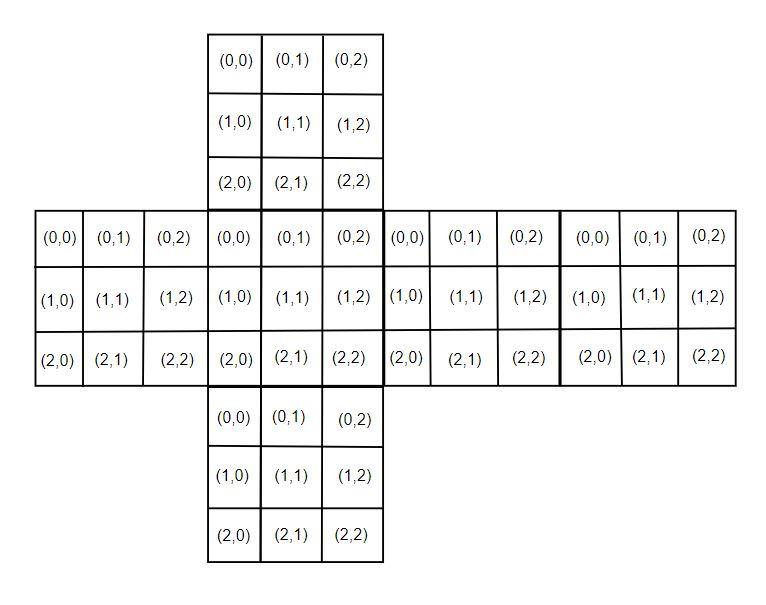
i. If a given face is rotated by 90 degrees clockwise about the axis passing from the centre of the face to the centre of the cube, the move is denoted by the first letter of the name of the face.

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

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

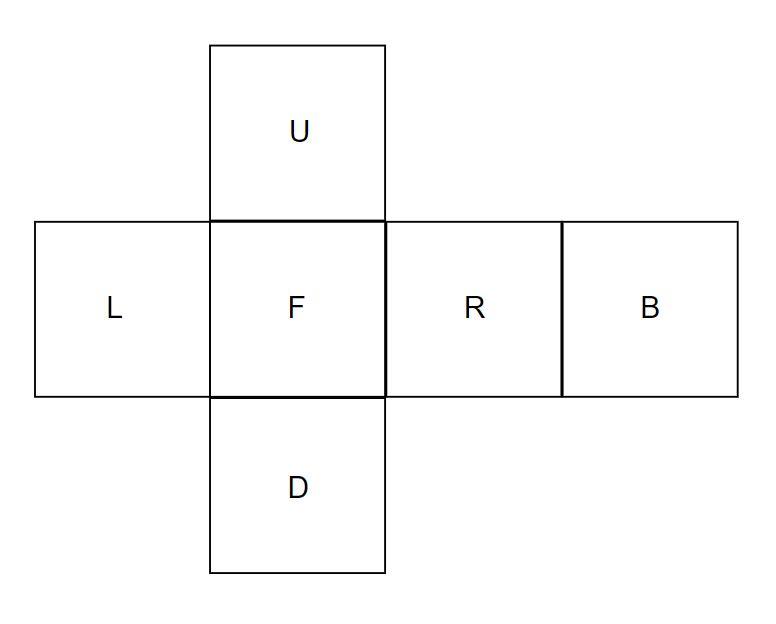
3. Any given face of the cube is a 3x3 square matrix whose indices are denoted by (0,0) to (2,2). Diagram below illustrates the same.

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

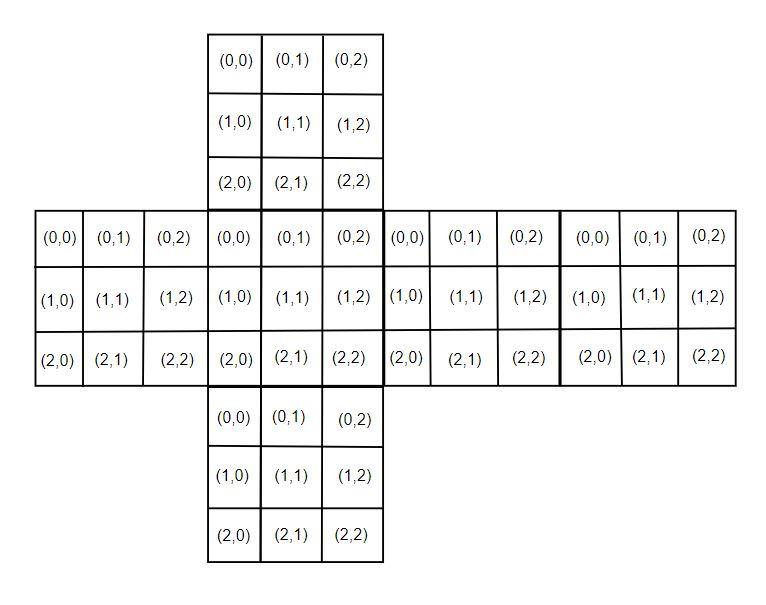
i. If a given face is rotated by 90 degrees clockwise about the axis passing from the centre of the face to the centre of the cube, the move is denoted by the first letter of the name of the face.

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

Maximum Prize

Problem Description

Imagine you are a martial arts fighter fighting with fellow martial artists to win prize money. However unlike traditional competitions, here you have the opportunity to pick and choose your opponent to maximize your prize kitty. The rules of maximization of prize kitty are as follows

► You have a superpower bestowed upon you, that you will win against anyone you challenge

► You have to choose the right order because unfortunately the superpower does not ensure that your prize money is always the highest

► Every victory against an opponent that you challenge and win against, will translate into a certain winning sum

► Here begins the technical part that you need to know in order to maximize your winning prize money

* All your opponents are standing in one line next to each other i.e. the order of opponents is fixed
* Your first task is to choose a suitable opponent from this line
* When you choose one opponent from that line, he steps out of the line and fights you.
* After you beat him, you get to decide how your prize money for winning against him will be calculated
* Essentially, if the opponent you have beaten has two neighbours, then you have the option to multiply the *opponent number* with any one of the two neighbours and add the other *opponent number.* That value becomes your prize money for that match
* If your opponent has only one neighbor then your prize money for that match is product of current *opponent number* with neighbours' *opponent number*
* When dealing with last opponent in the tournament, your prize money is equal to the value of the last *opponent number*
* As the tournament proceeds, the opponent that you have beaten has to leave the tournament

Example: 2 5 6 7

This depicts that you have four opponents with numbers 2 5 6 and 7 respectively

    1. Suppose you choose to fight opponent number 5, then after winning, the max prize kitty you can win for that match is = 5\*6+2 = 32

        Now opponent number 5 is out of the game. So opponent number 2 6 7 remain

    2. Suppose you now choose to fight opponent number 2, then after winning, the max prize kitty you can win for that match is = 2\*6+0 = 12. Your overall prize kitty is now 32 + 12 = 44

        Now opponent number 2 is out of the game. So opponent number 6 7 remain

    3. Suppose you now choose to fight opponent number 6, then after winning, the max prize kitty you can win for that match is = 7\*6+0 = 42. Your overall prize kitty is now 44 + 42 = 86

        Now opponent number 6 is out of the game. So opponent number 7 remains

    4. After beating opponent number 7, the max prize kitty you can win for that match is 7

        So overall prize kitty in this case is 93.

Other orders of choosing opponents will yield the following overall prize kitty

* Order 7->2->6->5 will yield overall prize kitty as 87
* Order 2->5->6->7 will yield overall prize kitty as 88
* Order 5->6->2->7 will yield overall prize kitty as 95
* Order 6- >7->2->5 will yield overall prize kitty as 97
* But by following the order 6->5->2->7 will yield overall prize kitty as 105, which is maximum.

Your task is to maximize your prize kitty by taking the right decisions

Input

First line contains an integer N which denotes the number of opponents in the tournament

Second line contains N space separated integers, which are the *opponent numbers*of other opponents

Output

Print the maximum number of coins you can win

Constraints

1 <= N <= 500

0 <= individual coin count < 100

Time Limit

1

Examples

Example 1

Input

4

2 5 6 7

Output

105

Explanation:

Refer the explanation in problem description.

Example 2

Input

3

7 8 9

Output

151

Explanation:

    1. You choose to fight opponent number 8, then after winning, the max prize kitty you can win for that match is = 8\*9+7 = 79

    Now opponent number 8 is out of the game. So opponent number 7 9 remain

    2. Suppose you now choose to fight opponent number 7, then after winning, the max prize kitty you can win for that match is = 7\*9+0 = 63. Your overall prize kitty is now 79 + 63 = 142

    Now opponent number 7 is out of the game. So opponent number 9 remains

    3. After beating opponent number 9, the max prize kitty you can win for that match is 9

    So overall prize kitty in this case is 142 + 9 = 151.