Australian Informatics Olympiad
Thursday 26 August, 2021

Intermediate Paper

Duration: 3 hours

6 questions

All questions should be attempted

Problem 1
Robot Vacuum

Input File: `robotin.txt`
Output File: `robotout.txt`

Time and Memory Limits: 1 second, 1 GB

Irene has a robot vacuum cleaner that automatically cleans her floor. Each day, the robot performs the same sequence of **K** instructions.

There are four possible instructions, each represented by an uppercase character. Each instruction moves the robot one *step* in one of the four cardinal directions:

- `N` - the robot moves one step north (↑).
- `E` - the robot moves one step east (→).
- `S` - the robot moves one step south (↓).
- `W` - the robot moves one step west (←).

After the sequence of **K** instructions, the robot is supposed to finish where it started, but the original programmers were a little bit rushed.

Irene has asked for your programming help. What is the smallest number of instructions needed to add to the end of the sequence so that the robot finishes where it started?

Input

- The first line of input contains the single integer **K**.
- The second line of input contains a string of **K** characters, the sequence of instructions.

Output

Your program should output a single integer, the fewest instructions you need to add to the end of the sequence.

Sample Input 1

```
5
ENNEE
```

Sample Output 1

```
5
```

Sample Input 2

```
7
EEWWWWE
```

Sample Output 2

```
1
```

Sample Input 3

```
8
SWWNENNS
```

Sample Output 3

```
2
```

Sample Input 4

```
4
NESW
```

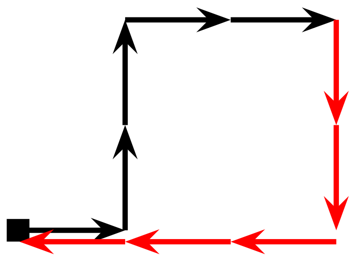
Sample Output 4

```
0
```

Explanation

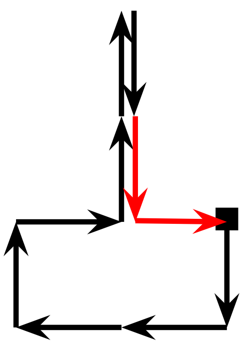
Each of the sample cases are illustrated below. The starting point is marked by a square, the original instructions are marked by black arrows and instructions added to the sequence are marked by red arrows.

Note that in the fourth sample input, no instructions need to be added, so the answer is 0.

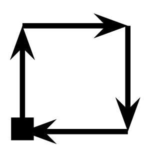


Sample Input 1

Sample Input 2



Sample Input 3



Sample Input 4

Subtasks & Constraints

For all test cases:

- $\mathbf{1 \le K \le 100\,000}$.

Additionally:

- For Subtask 1 (35 points), the initial $\mathbf{K}$ instructions only move the robot north (↑) and east (→). However, you can use any of the four possible instructions to return the robot to the start.
- For Subtask 2 (35 points), the $\mathbf{K}$ instructions only move the robot east (→) and west (←). However, you can use any of the four possible instructions to return the robot to the start.
- For Subtask 3 (30 points), no special constraints apply.

Problem 2
Art Class II

Input File: `artin.txt`
Output File: `artout.txt`

Time and Memory Limits: 1 second, 1 GB

You and your classmates were being a little too rowdy during art class this morning and have made a bit of a mess.

There's a large, usually blank wall at the back of the classroom. Unfortunately, the class (mostly you, honestly) have made $\mathbf{N}$ holes in the wall. The $\mathbf{i}$th hole is located $\mathbf{x_i}$ centimetres from the left edge of the wall, and $\mathbf{y_i}$ centimetres from the bottom edge of the wall.

You and your class have decided to create a single large poster to put on the wall to cover the holes.

To be as inconspicuous as possible, your poster should be:

- rectangular in shape
- hung so its sides are parallel to the edges of the wall
- large enough to cover all of the holes.

Note that a hole on the very edge of the poster will still be hidden (seems like your teacher forgot their glasses today).

You don't have a lot of time before the teacher returns, so you've decided to write a program that will tell you the area of the smallest poster that will cover all the holes.

Input

- The first line of input contains the single integer $\mathbf{N}$.
- The next $\mathbf{N}$ lines describe the location of the holes. The $\mathbf{i}$th line contains the two integers $\mathbf{x_i}$ and $\mathbf{y_i}$.

Output

Your program should output a single integer, the area (in square centimetres) of the smallest poster that will cover all the holes.

Sample Input 1

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

Sample Input 2

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

Sample Input 3

```
2
2 1
3 1
```

Sample Output 1

```
12
```

Sample Output 2

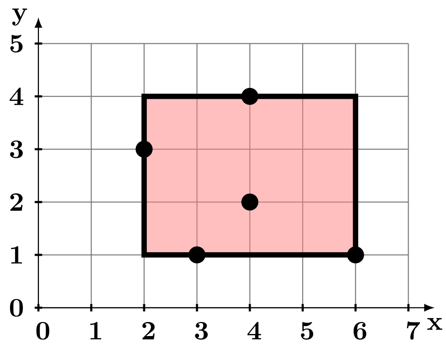
```
16
```

Sample Output 3

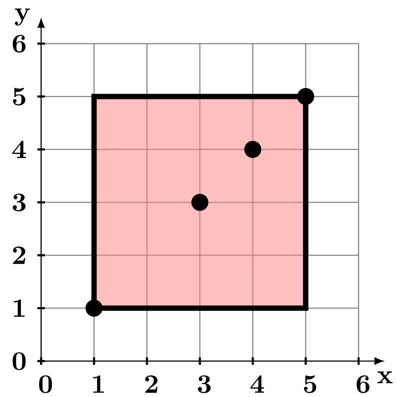
```
0
```

Explanation

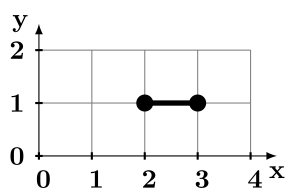
Each of the sample cases are illustrated below. Note that in the third sample input, the holes can be covered by an infinitely thin poster with a height of zero. The area of this poster is zero.



Sample Input 1



Sample Input 2



Sample Input 3

Subtasks & Constraints

For all test cases:

- $\mathbf{2 \le N \le 100\,000}$.
- $\mathbf{1 \le x_i \le 10\,000}$, for all $\mathbf{i}$.
- $\mathbf{1 \le y_i \le 10\,000}$, for all $\mathbf{i}$.

Additionally:

- For Subtask 1 (40 points), $\mathbf{N = 2}$.
- For Subtask 2 (40 points), $\mathbf{x_i = y_i}$, for all $\mathbf{i}$. See Sample Input 2 for an example.
- For Subtask 3 (20 points), no special constraints apply.

Problem 3
Melody

Input File: `melodyin.txt`
Output File: `melodyout.txt`

Time and Memory Limits: 1 second, 1 GB

Your friend Melody has written a song. Melody's song is a sequence of $\mathbf{N}$ notes, where each note is represented by an integer between $\mathbf{1}$ and $\mathbf{K}$. **$\mathbf{N}$ is always a multiple of three.**

You happen to be very knowledgeable in the trends of modern pop music, and know that a song is *nice* if and only if it consists of the *same sequence of three notes repeated over and over again*, until the end.

For example, the following songs are nice:

- $\mathbf{1, 2, 3, 1, 2, 3, 1, 2, 3}$ (since $\mathbf{1, 2, 3}$ is repeated over and over again)
- $\mathbf{6, 4, 4, 6, 4, 4}$ (since $\mathbf{6, 4, 4}$ is repeated over and over again)
- $\mathbf{8, 8, 8, 8, 8, 8}$ (since $\mathbf{8, 8, 8}$ is repeated over and over again)
- $\mathbf{6, 2, 6}$ (since $\mathbf{6, 2, 6}$ is repeated over and over again).

The following songs are not nice:

- $\mathbf{1, 2, 3, 4, 5, 6}$
- $\mathbf{8, 8, 8, 8, 8, 4}$
- $\mathbf{1, 2, 3, 1, 3, 2, 1, 2, 3}$
- $\mathbf{2, 2, 5, 5, 2, 2}$.

After hearing your advice, Melody would like to change (but not add or remove) some of, or none of, the notes in her song, so that it is nice.

What is the smallest possible number of notes she can change so that her song is nice?

Input

- The first line of input contains the two integers $\mathbf{N}$ and $\mathbf{K}$.
- The next $\mathbf{N}$ lines each contain one integer. The $\mathbf{i}$th of these is $\mathbf{S_i}$, the $\mathbf{i}$th note in Melody's song.

Output

Your program should output a single integer, the smallest possible number of notes Melody can change so that her song is nice.

Sample Input 1

```
6 4
1
2
3
4
3
3
```

Sample Input 2

```
9 10
8
8
3
8
8
3
4
8
3
```

Sample Input 3

```
3 2
2
2
1
```

Sample Output 1

```
2
```

Sample Output 2

```
1
```

Sample Output 3

```
0
```

Explanation

In the first sample input, the song $\mathbf{1, 2, 3, 4, 3, 3}$ could be made into the nice song $\mathbf{1, 2, 3, 1, 2, 3}$. This requires 2 notes (those in the 4th and 5th positions) to be changed, which is the fewest possible.

In the second sample input, the song $\mathbf{8, 8, 3, 8, 8, 3, 4, 8, 3}$ could be made into the nice song $\mathbf{8, 8, 3, 8, 8, 3, 8, 8, 3}$. This requires 1 note (the one in the 7th position) to be changed, which is the fewest possible.

In the third sample input, the song $\mathbf{2, 2, 1}$ is already nice, so no notes need to be changed.

Subtasks & Constraints

For all test cases:

- $\mathbf{3 \le N \le 99\,999}$.
- $\mathbf{N}$ is a multiple of three.
- $\mathbf{1 \le K \le 100\,000}$.
- $\mathbf{1 \le S_i \le K}$, for all $\mathbf{i}$.

Additionally:

- For Subtask 1 (20 points), making Melody's song into the nice song $\mathbf{1, 2, 3, 1, 2, 3}, \ldots$ will require the smallest possible number of changes. See Sample Input 1 for an example.
- For Subtask 2 (30 points), $\mathbf{N \le 30}$ and $\mathbf{K \le 30}$.
- For Subtask 3 (50 points), no special constraints apply.

Problem 4
Social Distancing

Input File: `distin.txt`
Output File: `distout.txt`

Time and Memory Limits: 1 second, 1 GB

The hippopotami of North Yorkshire are hosting their annual Grand Banquet tonight!

The banquet is an ancient tradition, and is hosted by the hippo mayor on the main street of North Yorkshire. In a normal year, the mayor invites all $\mathbf{N}$ hippos to the event, and each of them is provided with a meal consisting of a mouth-watering main course followed by an enormous dessert. The head chef prepares the $\mathbf{N}$ meals, the $\mathbf{i}$th of which is placed $\mathbf{D_i}$ metres along the street for a hippo to eat.

However, due to social distancing requirements, each hippo is required to stay at least $\mathbf{K}$ metres away from every other hippo at all times. This means that the $\mathbf{i}$th and $\mathbf{j}$th meal cannot both be eaten if they are strictly less than $\mathbf{K}$ metres away from each other. If they are exactly $\mathbf{K}$ or more metres away from each other, then they can both be eaten.

The mayor wishes to invite as many hippos as possible, while still obeying the social distancing requirements. As the head informatician for the hippos, the mayor has turned to you for help. What is the maximum number of hippos that can be invited?

Input

- The first line of input contains the two integers $\mathbf{N}$ and $\mathbf{K}$.
- The next $\mathbf{N}$ lines each contain one integer. The $\mathbf{i}$th of these is $\mathbf{D_i}$, the location of the $\mathbf{i}$th meal.

Output

Your program should output a single integer, the maximum number of hippos that can be invited.

Sample Input 1

```
3 1
3
2
3
```

Sample Input 2

```
3 2
1
3
7
```

Sample Input 3

```
5 4
3
5
1
12
10
```

Sample Output 1

```
2
```

Sample Output 2

```
3
```

Sample Output 3

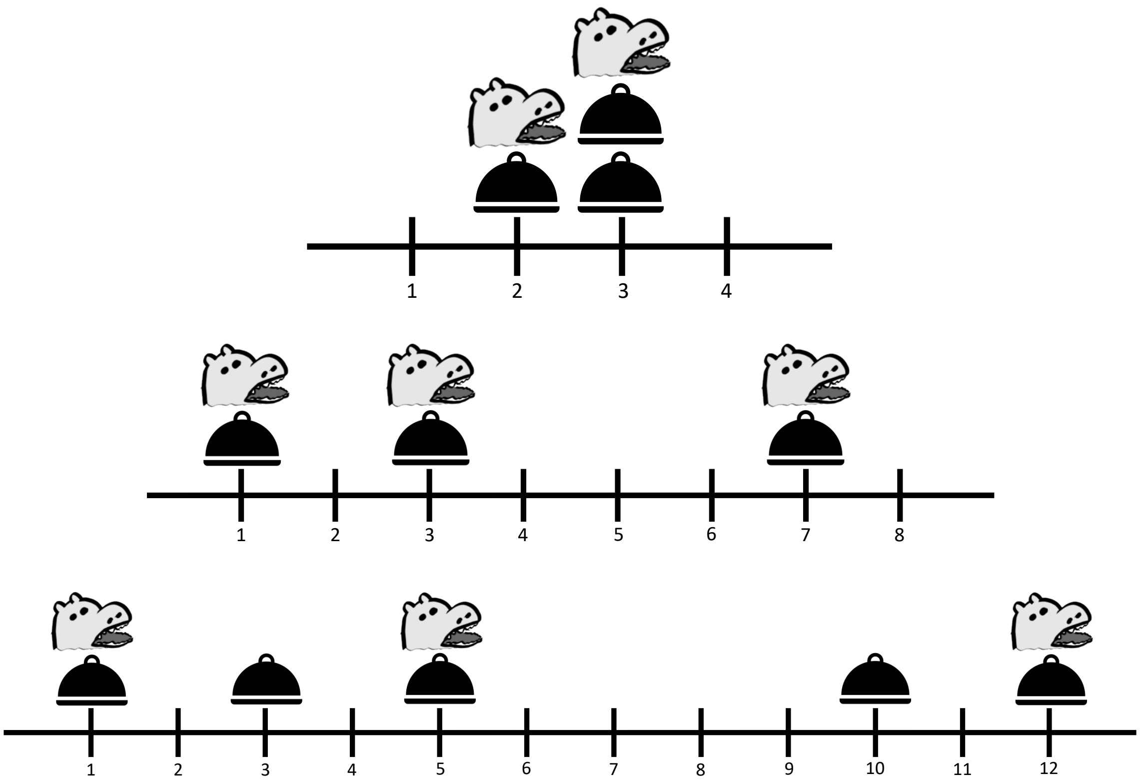
```
3
```

Explanation

Each of the samples are shown below with one possible solution. The dishes represent the meals, and hippos are shown above the meals which can be eaten.

In the first sample input there are 2 meals with $\mathbf{D_i = 3}$, but only one hippo can be invited to eat them due to social distancing requirements. Another hippo can be invited to eat the meal at $\mathbf{D_i = 2}$, because they are $\mathbf{K = 1}$ metres away from the other hippo.

In the third sample input there is another solution which also achieves the maximum number of hippos that can be invited.



Subtasks & Constraints

For all test cases:

- $\mathbf{2 \leq N \leq 100\,000}$.
- $\mathbf{1 \leq K \leq 1\,000\,000\,000}$.
- $\mathbf{1 \leq D_i \leq 1\,000\,000\,000}$ for all $\mathbf{i}$.

Additionally:

- For Subtask 1 (15 points), $\mathbf{N = 2}$.
- For Subtask 2 (25 points), $\mathbf{K = 1}$.
- For Subtask 3 (45 points), $\mathbf{N \leq 1000}$.
- For Subtask 4 (15 points), no special constraints apply.

Problem 5
Space Mission

Input File: `spacein.txt`
Output File: `spaceout.txt`

Time and Memory Limits: 1 second, 1 GB

The Australian Interstellar Operations Council is planning a space mission to collect gas samples from the nearby star *Albert Major*. Fortunately, recent advancements in spacefaring technology allow practically instant travel to and from the star using single-use portals. Time is limited however, as the star is predicted to turn supernova very soon.

The council is planning to send a probe to retrieve samples some time in the next $\mathbf{N}$ days (numbered from 1 to $\mathbf{N}$). You have been tasked with choosing the *launch day* and the *return day* for the probe. **The launch day cannot be the same day as the return day**.

Due to constant fluctuations in the fabric of space-time, it requires $\mathbf{C_i}$ units of fuel to use a portal on day $\mathbf{i}$ (whether launching or returning).

Each day (including the launch and return days), the probe collects one gas sample from the star. You'd like to collect as many samples as possible, however the council **only has F units of fuel to use in total**.

What is the maximum number of samples you could collect?

Input

- The first line of input contains the two integers $\mathbf{N}$ and $\mathbf{F}$.
- The next $\mathbf{N}$ lines each contain one integer. The $\mathbf{i}$th of these is $\mathbf{C_i}$, the cost to open a portal on the $\mathbf{i}$th day.

Output

Your program should output a single integer, the maximum number of samples you could collect. **If there is not enough fuel to choose a launch and return day, print `-1` instead.**

Sample Input 1

```
5 50
10
40
20
30
75
```

Sample Input 2

```
7 14
12
8
2
16
4
6
10
```

Sample Input 3

```
2 4
4
1
```

Sample Output 1

```
4
```

Sample Output 2

```
5
```

Sample Output 3

```
-1
```

Explanation

In the first sample input, the council has $\mathbf{F} = \mathbf{50}$ units of fuel. If you launch on day 1 and return on day 4, you will collect 4 samples, costing $\mathbf{10} + \mathbf{30} = \mathbf{40}$ units of fuel. This is the maximum number of samples you can collect.

Day	1	2	3	4	5
Fuel	10	40	20	30	75

In the second sample input, the council has $\mathbf{F} = \mathbf{14}$ units of fuel. If you launch on day 2 and return on day 6, you will collect 5 samples, costing $\mathbf{8} + \mathbf{6} = \mathbf{14}$ units of fuel. This is the maximum number of samples you can collect.

Day	1	2	3	4	5	6	7
Fuel	12	8	2	16	4	6	10

In the third sample input, there is not enough fuel to launch the probe and have it return on a different day, so the answer is $-\mathbf{1}$.

Day	1	2
Fuel	4	1

Subtasks & Constraints

For all test cases:

- $\mathbf{2} \leq \mathbf{N} \leq \mathbf{100\,000}$.
- $\mathbf{1} \leq \mathbf{F} \leq \mathbf{1\,000\,000\,000}$.
- $\mathbf{1} \leq \mathbf{C_i} \leq \mathbf{1\,000\,000\,000}$ for all $\mathbf{i}$.

Additionally:

- For Subtask 1 (20 points), $\mathbf{N} \leq \mathbf{1000}$.
- For Subtask 2 (50 points), $\mathbf{F} \leq \mathbf{100}$.
- For Subtask 3 (30 points), no special constraints apply.

Problem 6
Laser Cutter

Input File: `laserin.txt`
Output File: `laserout.txt`

Time and Memory Limits: 1 second, 1 GB

Cameron has a square piece of plywood **N** centimetres tall by **N** centimetres wide. He has a robotic laser cutter that can make cuts starting in the top-left corner and ending in the bottom-right corner.

To use the laser cutter, Cameron programs the robot with a sequence of **2N** instructions. There are two possible instructions, each represented by an uppercase character:

- `R` - the laser makes a cut by moving one centimetre to the right.
- `D` - the laser makes a cut by moving one centimetre down.

Cameron first gave the robot the sequence of instructions **A**, to cut out the lower boundary of the shape. Then, he gave the robot the sequence **B**, to cut out the upper boundary of the shape. *The two cuts did not intersect, except in the top-left and bottom-right corners.*

Help Cameron find *the side length of the largest square* that fits inside the shape that he cut out. The sides of the square must be parallel to the sides of the piece of plywood.

Input

- The first line of input contains the integer **N**.
- The second line of input contains the sequence of instructions **A**, as a string of **2N** characters.
- The third line of input contains the sequence of instructions **B**, as a string of **2N** characters.

Output

Your program should output a single integer, the side length of the largest square that fits inside the shape Cameron cut out.

Sample Input 1

```
10
DDDDRRDDDDRRRDRDRRRR
RRDRDRRRDDRRDDDDRDRD
```

Sample Output 1

```
4
```

Sample Input 2

```
6
DDDDDDRRRRRR
RDRRDDDRRRDD
```

Sample Output 2

```
3
```

Sample Input 3

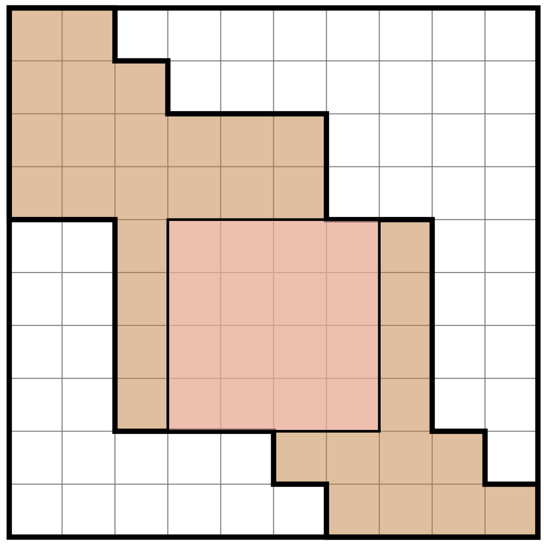
```
5
DDRRDDRDRR
RDRRDRDRDD
```

Sample Output 3

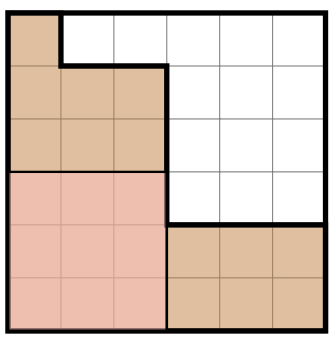
```
2
```

Explanation

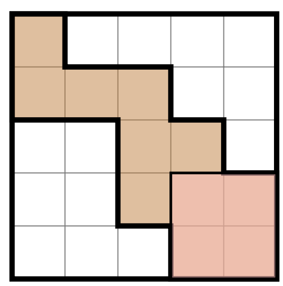
Each of the samples are shown below with one possible largest square.



Sample Input 1



Sample Input 2



Sample Input 3

Subtasks & Constraints

For all test cases:

- $\mathbf{2 \leq N \leq 200\,000}$.

Additionally:

- For Subtask 1 (28 marks), The bottom-left corner is always part of the cut out shape. See Sample Input 2 for an example.
- For Subtask 2 (16 marks), $\mathbf{N \leq 10}$.
- For Subtask 3 (26 marks), $\mathbf{N \leq 1000}$.
- For Subtask 4 (30 marks), no special constraints apply.