Nordic Collegiate Programming Contest

NCPC 2024

October 5, 2024

Problems

A Avoiding the Abyss
B Baseball Court
C Composed Rhythms
D Double Deck
E Elapid Errands
F Fence Fee
G Guessing Passwords
H Hotfix
I Infinite Cash
J Jungle Game
K Knitting Pattern

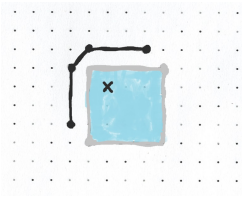
Do not open before the contest has started.

Advice, hints, and general information

- The problems are **not** sorted by difficulty.
- Your solution programs must read input from *standard input* (e.g. `System.in` in Java or `cin` in C++) and write output to *standard output* (e.g. `System.out` in Java or `cout` in C++). For further details and examples, please refer to the documentation in the help pages for your favorite language on Kattis.
- For information about which compiler flags and versions are used, please refer to the documentation in the help pages for your favorite language on Kattis.
- Your submissions will be run multiple times, on several different inputs. If your submission is incorrect, the error message you get will be the error exhibited on the first input on which you failed. E.g., if your instance is prone to crash but also incorrect, your submission may be judged as either "Wrong Answer" or "Run Time Error", depending on which is discovered first. The inputs for a problem will always be tested in the same order.
- If you think some problem is ambiguous or underspecified, you may ask the judges for a clarification request through the Kattis system. The most likely response is "No comment, read problem statement", indicating that the answer can be deduced by carefully reading the problem statement or by checking the sample test cases given in the problem, or that the answer to the question is simply irrelevant to solving the problem.
- In general we are lenient with small formatting errors in the output, in particular whitespace errors within reason, and upper/lower case errors are often (but not always) ignored. But not printing any spaces at all (e.g. missing the space in the string "`1 2`" so that it becomes "`12`") is typically not accepted. The safest way to get accepted is to follow the output format exactly.
- For problems with floating point output, we only require that your output is correct up to some error tolerance. For example, if the problem requires the output to be within either absolute or relative error of 10^{-4}, this means that
  - If the correct answer is 0.05, any answer between 0.0499 and $.0501$ will be accepted.
  - If the correct answer is 500, any answer between 499.95 and 500.05 will be accepted.

  Any reasonable format for floating point numbers is acceptable. For instance, "17.000000", "0.17e2", and "17" are all acceptable ways of formatting the number 17. For the definition of reasonable, please use your common sense.

Problem A
Avoiding the Abyss



This picture represents sample 1. The path taken avoids the hidden pool, but based on the information given it could also have intersected it. So the sample solution was quite lucky here.

You are standing on a point with integer coordinates (x_s, y_s). You want to walk to the point with integer coordinates (x_t, y_t). To do this, you can walk along a sequence of line segments. But there is a swimming pool in your way. The swimming pool is an axis aligned rectangle whose lower left corner is on the point (x_l, y_l) and the upper right corner is on the point (x_r, y_r). You cannot ever cross the swimming pool, not even on the border. However, it is dark and you do not know the coordinates (x_l, y_l) and (x_r, y_r). Instead, you threw a rock into the pool which revealed that the point (x_p, y_p) is in the pool (or on the boundary).

Find a way to walk from the start to the end point along a sequence of line segments, so that you never cross the swimming pool.

Input

The first line contains two integers x_s and y_s ($-10^4 \leq x_s, y_s \leq 10^4$).

The second line contains two integers x_t and y_t ($-10^4 \leq x_t, y_t \leq 10^4$).

The third line contains two integers x_p and y_p ($-10^4 \leq x_p, y_p \leq 10^4$).

The problem is not adaptive, i.e. for every test case there exist four integers x_l, y_l, x_r, y_r ($-10^4 \leq x_l < x_r \leq 10^4$, $-10^4 \leq y_l < y_r \leq 10^4$) that constitute a swimming pool. The start and end points are always strictly outside the swimming pool, and the point (x_p, y_p) is inside (or on the border). The start and end points are always distinct.

Output

First, print one integer N ($0 \leq N \leq 10$), the number of points in between the start and end point that you want to visit. Then, print N lines, the ith containing two integers x_i, y_i. These coordinates must satisfy $-10^9 \leq x_i, y_i \leq 10^9$. Note that these are not the same bounds than on the other coordinates.

This means that you will walk along straight line segments between $(x_s, y_s), (x_1, y_1), \ldots, (x_N, y_N), (x_t, y_t)$ such that none of the line segments touch the swimming pool. It can be proven that a solution always exists.

Sample Input 1	Sample Output 1
0 0 4 4 2 2	2 0 3 1 4

This page is intentionally left blank.

Problem B
Baseball Court

Image taken from commons.wikimedia.org.

After lengthy meetings on the subject, the NCPC jury has decided that getting the contestants' blood pumping would result in better contests. Thus they figured it would be a good idea to compete for bragging rights in some kind of sport prior to the programming contest. For this they need some place to compete and to pick a sport to compete in. One of those is solved easily by drawing at random from a hat, which results in the chosen sport being baseball. That simply leaves the matter of making a suitable court to play on. The NCPC jury has access to a rectangular plot of land a by b meters in size. Furthermore, they have N square tiles of grass they can place on this plot to create the court. All grass tiles must be placed with their sides parallel to the edges of the plot.

The south-west corner of the land is chosen to be the batting point. For the placement of the grass tiles to constitute a valid court, two conditions must be met. Firstly, for any given tile the south and west sides must either lay directly against the edge of the plot or directly against another tile. This is required to make sure the ball doesn't exit and reenter the court while following a straight trajectory. Secondly, all tiles which have no adjacent tile to the north nor to the east must have the same manhattan distance from the batting point. This is to prevent batters from preferring some directions over others.

Your task is to find the number of ways to place the tiles so that it creates a valid baseball court.

Input

The first line of the input contains a single positive integer N ($1 \leq N \leq 10^4$), the number of grass tiles available. The second line of the input contains two positive integers a and b ($1 \leq a, b \leq 10^4$), the size of the land the court can be made within.

Output

Print the number of valid ways to place the grass tiles to make up a baseball court. All n grass tiles must be used. Since this number might be very large, print the answer modulo $10^9 + 7$.

Sample Input 1	Sample Output 1
15 3 8	3

Sample Input 2	Sample Output 2
15 3 5	1

Sample Input 3	Sample Output 3
15 3 4	0

Problem C
Composed Rhythms

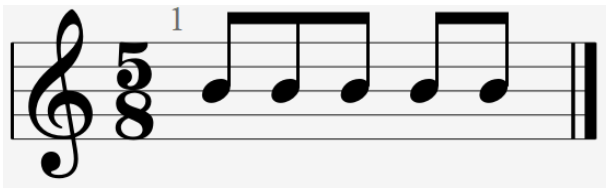


Image by Gunnlaugur Arnarson

Rhythm is an important part of music and it is crucial for aspiring musicians to gain understanding of it. As the skill of the musician advances, more complex rhythms are introduced to them. To ease the learning of musical passages, a method of simplifying rhythms can be helpful. One method is to reduce the rhythm into groups of twos and threes.

A rhythm is composed of multiple beats. A single beat does not make up a rhythm, as the beats depend on each other. The rhythm can be subdivided into smaller components. For example, a rhythm of 7 beats can be subdivided into 4 beats and 3 beats, or alternatively into 2, 3, and 2 beats. However, a rhythm of 7 beats cannot be subdivided into 1, 3, and 3 beats, since one of the components does not form a rhythm.

This leaves 2 as the smallest group size of beats we can use to decompose a rhythm, but if we only use groups of size 2 then we cannot have an odd number of beats. Adding 3 as a group size allows us to decompose any rhythm, even if it has an odd number of beats.

Given the number of beats in a rhythm, provide one decomposition of the rhythm into groups of sizes 2 and 3.

Input

The first and only line of input contains a single integer N ($2 \leq N \leq 10^6$), denoting the number of beats in the rhythm.

Output

First output one line with an integer K, the number of groups of which your decomposition consists. Then output a line with K space-separated integers, each of which is a 2 or a 3. Your decomposition must be made up of the correct number of beats.

If there are multiple correct answers, you may output any of them.

Sample Input 1	Sample Output 1
25	9 3 3 3 3 3 3 2 2 3

This page is intentionally left blank.

Problem D
Double Deck

Image taken from wikimedia.org.

You are playing a new card game. In the game you have two decks of cards each consisting of $N \cdot K$ cards labeled with an integer from 1 to N, inclusive. Also, each type of card appears precisely K times in each deck.

The rules of the game are simple. You shuffle both decks and place them face up in front of you, so at each point in time you see the top card in each deck. If the top cards are the same you can take them both and get one point. Otherwise you must discard either card. Your goal is to get as many points as possible.

You have just finished playing a round of this game and you want to know what the maximum score was, knowing the layout of both decks.

Input

The first line of the input contains two integers N and K ($1 \leq N \leq 10^4, 1 \leq K \leq 15$). The second and third line of the input each contain $N \cdot K$ integers x_i ($1 \leq x_i \leq N$), describing the layout of the decks. The first number x_1 is the topmost card in the deck, x_2 is the second, and so on.

No integer in the second line and third line is repeated more than K times per line.

Output

Print a single integer, the maximum possible score.

Sample Input 1

```
3 2
3 1 2 3 1 2
2 1 3 1 3 2
```

Sample Output 1

```
4
```

Sample Input 2

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

Sample Output 2

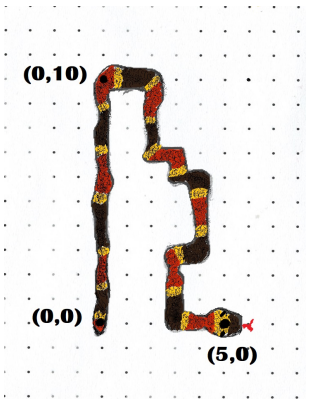
```
8
```

This page is intentionally left blank.

Problem E
Elapid Errands

Carl the snake is in his burrow at the point $(0, 0)$ in an infinite plane. He wants to visit the points $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$. These points must be visited in the order they are given, and Carl must end up at the point (x_N, y_N). In one move, he can move one step up, down, left, or right. However, since he is a very long snake, he can never visit the same point more than once.

Your task is to find a sequence of moves such that Carl visits all the points in order, and never visits any point more than once. The points (x_i, y_i) were generated uniformly at random.



The journey of the snake in the sample.

Input

The first line contains one integer N ($1 \leq N \leq 20$), the number of points you must visit.

The following N lines each contain two integers x_i, y_i ($0 \leq x_i, y_i \leq 10^4$).

Apart from the sample, there will be 100 testcases, all with $N = 20$. The (manhattan) distance between any two of the points (including the starting point $(0, 0)$) will be at least 20. Within these constraints, the points (x_i, y_i) were generated uniformly at random.

Note that the sample does not satisfy the distance requirement. Your solution **does not** need to solve the sample to get accepted.

Output

Print a string consisting of the characters '<', '>', '^', 'v'. This is the list of moves you should make so that you visit all the points in order without ever going to the same point more than once. The string must have length at most $2 \cdot 10^6$.

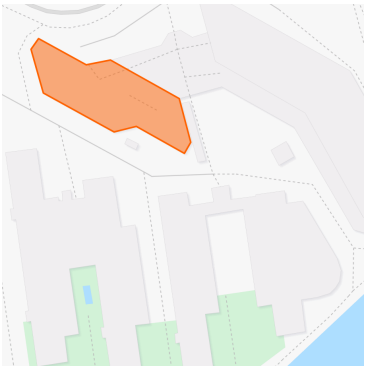
Sample Input 1	Sample Output 1
2 0 10 5 0	^^^^^^^^^^>>vvv>v>vvv<vvv>>

This page is intentionally left blank.

Problem F
Fence Fee

The National Crop Protection Commission (NCPC) is dedicated to supporting local farmers by offering subsidies that are proportional to the area of their crop fields. Each farmer can have several crop fields, each uniquely shaped but geometrically defined as a polygon, bounded by fences that meet only at the corners.

In a classic display of political bureaucracy, and to incentivize well-shaped fields, the NCPC will subsidize each crop field based on the square of its area. This, to reward well-encapsulated fields. They now require a tool that calculates the sum of the squared areas of these polygons to ensure that the subsidies are distributed fairly.



Generated from OpenStreetMap, ODBL 1.0

Input

The first line contains an integer F ($3 \leq F \leq 1000$), the number of fence line segments. The next F lines contain four integers each, x_1, y_1, x_2, y_2 ($0 \leq x_1, y_1, x_2, y_2 \leq 1000$), representing a straight-line fence section.

No two fence line segments intersect. Since fencing is both expensive and tedious, you may assume that every fence line segment is necessary and serves to bound fields. All fences farmers have are connected.

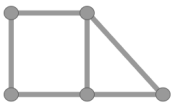
In other words, the graph that consists of endpoints and fences is planar, connected, and has no bridges.

Output

Output a single line representing the sum of the squared areas of all the fields formed by the given fence sections. Your answer will be correct if it has an absolute or relative error of at most 10^{-6}.

Explanation of sample

The picture represents sample 2. The areas of the two fields are 1 and 0.5, so the sum of their squares is $1 + 0.25 = 1.25$.



Sample Input 1	Sample Output 1
5 0 0 0 1 0 1 1 1 0 0 1 0 1 0 2 0 1 1 2 0	2.25

Sample Input 2	Sample Output 2
6 0 0 0 1 0 1 1 1 0 0 1 0 1 0 2 0 1 1 2 0 1 0 1 1	1.25

Problem G
Guessing Passwords

Ingfríður is testing her new pet project website, Passwordle. The rules should be rather familiar for those who have played plenty of wordle, but let us review them here. The website picks a secret password that the user then has to guess. The password will be N characters in length and the user will guess until they get it right, getting a higher score for fewer guesses. Each guess must be a string of N characters. For each character in the guess, it will be coloured one of three colours. If that character matches the password character in the same position, the colour is green. If the character does not match, but that character is in the password somewhere else, the colour is yellow. Otherwise it is gray.

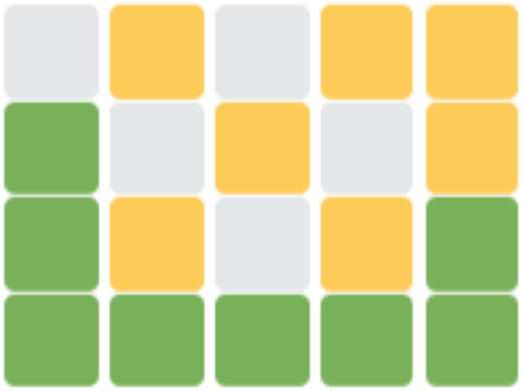


Image taken from commons.wikimedia.org.

Ingfríður is of course very good at her own game, so she will always guess a password that could be the hidden password. That is to say her guess will always match the previous clues. Furthermore she knows her program never generates passwords with repeated characters, since that's insecure, and will incorporate this knowledge into her guesses.

You now receive some screenshots from her testing progress, but they got so terribly compressed you can only make out the colours and not the text itself. Furthermore it seems that she didn't manage to make any progress at all. She didn't get a single green square in the entire game and never found any more characters than she did in her very first guess, and quit out of frustration. So the number of yellow squares is the same on each row. Given this info, can you reconstruct a sequence of guesses she could have made? Or is it impossible and her program must be wrong? You may assume that Ingfríður never makes any mistake when playing, only when programming.

Input

The first line of the input contains two positive integers N and M ($1 \leq N, M \leq 100$). N is the number of characters in the password and M is the number of guesses in the screenshot. Next there are N lines, each with M characters, the i-th of which gives the colours of the i-th guess. `G` denotes gray and `Y` denotes yellow. The number of `Y`s is constant across all lines. The characters are given without spaces between them. Finally there is a single line with a single positive integer Σ ($1 \leq \Sigma \leq 10^6$), giving the number of valid characters for the passwords.

Output

If there is no way to achieve the given colours print `Bugged!`. Otherwise print $N + 1$ lines with M numbers each, separated by spaces. The first N lines should give the colouring in the input if the number i denotes the i-th character in the alphabet. The final and $(N + 1)$-st line should give a secret word that could have resulted in this sequence of guesses being valid. If there are multiple possible solutions any one of them will be accepted.

Explanation of sample

In the first sample, there is a valid sequence of guesses. In the first move, Ingfríður guesses the word `3 4 2 1`, which reveals that the 1 and the 2 are somewhere in the hidden password but not at those positions. The next guess is `1 5 6 2` which is a valid second guess since it could have been the password, given the information from the first guess. An example of an invalid second guess would be `1 3 2 5`. This is invalid because Ingfríður already knows that the 2 is not on the third position, and that the 3 shouldn't occur in the word at all.

Sample Input 1	Sample Output 1
3 4 GGYY YGGY GYYG 26	3 4 2 1 1 5 6 2 7 2 1 8 2 1 9 10

Sample Input 2	Sample Output 2
4 5 GYGGY YGYGG GGYYG GYGGY 16	Bugged!

Problem H
Hotfix

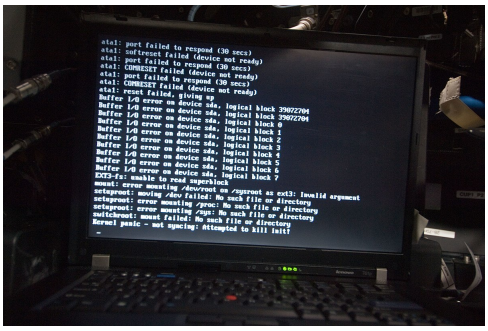


Image taken from commons.wikimedia.org.

In an earlier contest, contestants had to solve a simple problem. They were given a string and had to print each unique substring of it, along with the number of occurrences it had in the original string. For example `AB` would print `A 1 B 1 AB 1` and `AAA` would print `A 3 AA 2 AAA 1`.

When copying over this problem for reuse in this contest, several mistakes were made. The input constraints were changed significantly, making the problem absolutely impossible! Luckily this was partially cancelled out by the output validator being mangled as well. Now instead of checking for absolute correctness it only requires the number of occurrences of each character in the output to be correct. With a quick hotfix of applying run length encoding to the output, the problem was finally solvable again and the contest could continue on as planned. Right?

Input

The input contains a single string of length at least 1 and at most 10^6. It contains only ASCII upper and lower case characters. This string is then followed by a single newline character.

Output

For each non-whitespace character that appears a non-zero number of times in the output of the problem described above, print it along with its number of occurrences on a single line, separated by a space. Print the lines ordered by ascending values of the ASCII characters.

Sample Input 1	Sample Output 1

`ABC`	`1 6`
	`A 3`
	`B 4`
	`C 3`

Sample Input 2	Sample Output 2

`aaaab`	`1 6`
	`2 1`
	`3 1`
	`4 1`
	`a 20`
	`b 5`

This page is intentionally left blank.

Problem I
Infinite Cash

Image taken from flickr.com.

Svalur Handsome has finally graduated with a degree in computer science, and it couldn't have happened sooner. He has some rather unwise spending habits which he hopes will be more sustainable now that he can get a high paying job as a programmer. He has applied to a few places, and now has a contract in his hands that he could sign and start working almost immediately. But before he takes the offer he wants to figure out how long it could support his spending habits.

At the start of every day Svalur spends half of his remaining money, rounded up. The new job would pay s ISK at the end of every d-th day, starting with the d-th day. He currently has m ISK to spend as well.

Input

The input has three lines, each containing the positive integers s, d, m respectively. They satisfy $1 \leq s, d, m \leq 2^{1000}$. As these payment details are for a computer science job the numbers are all given in binary, naturally.

Output

Print the number of the day that Svalur wants to spend money, but has none. This should naturally also be printed in binary. If he can support his spending habits indefinitely instead print `Infinite money!`.

Sample Input 1	Sample Output 1
101110101 1010 10001110101010101	10011

Sample Input 2	Sample Output 2
101110101 1000 100011101	Infinite money!

Sample Input 3	Sample Output 3
101110101 1010 100011101	1001

This page is intentionally left blank.

Problem J
Jungle Game


Rainforest, public domain

Denise is designing a rainforest themed board game. The goal of the game is for each player to form a team of two characters and complete various challenges.

There are N different characters numbered from 1 to N. Each character i has two attributes p_i and s_i (problem solving skill and strength). The numbers p_i and s_i are positive integers satisfying $1 \leq p_i, s_i \leq N$. Before the game starts, each player will pick two characters i and j to form a team. It is possible to pick two copies of the same character. The total problem solving skill and strength of the team will be $p_i + p_j$ and $s_i + s_j$ respectively.

In the game there are also N challenge cards numbered from 1 to N. Each of these also has two attributes P_k and S_k. Denise has already designed the challenge cards and decided on the values of all numbers $P_1, P_2, \ldots, P_N$ and $S_1, S_2, \ldots, S_N$. However, the rules of the game assume that it is not possible for a player to form a team whose problem solving skill and strength are both the same as one of the challenge cards. In other words, the situation

$$p_i + p_j = P_k \text{ and } s_i + s_j = S_k$$

should never occur for any triple i, j, k (note that i can be equal to j).

The only thing left to do is to decide the N distinct pairs $(p_1, s_1), (p_2, s_2) \ldots, (p_N, s_N)$ such that $1 \leq p_i, s_i \leq N$ and the situation above never happens.

Input

The first line contains the integer N ($1 \leq N \leq 2000$).

The following N lines contain the values of the challenge cards P_i, S_i ($2 \leq P_i, S_i \leq 2 \cdot N$).

Output

If there is no solution, print "NO". Otherwise, print "YES" followed by N lines, each containing a pair of integers p_i, s_i ($1 \leq p_i, s_i \leq N$). These pairs of integers must be distinct. In other words, you may not have two indices $i \neq j$ with $p_i = p_j$ and $s_i = s_j$.

Sample Input 1

```
5
5 5
5 6
6 5
6 6
8 8
```

Sample Output 1

```
YES
2 2
1 1
1 2
2 1
1 3
```

Sample Input 2

```
1
2 2
```

Sample Output 2

```
NO
```

This page is intentionally left blank.

Problem K
Knitting Pattern

Jörmunrekur had found himself with some extra time on his hands, so he decided to try to find a new hobby. After discussing this with some of his relatives, his grandparents lent him a book with knitting guides and knitting patterns.

He wants to start with something big, so he decides to make a sweater. He has also picked out a pattern from the book that he will repeat around the circumference of the sweater. He wants the pattern to be centered and then repeat out towards the back in either direction, but never wants to have less than the full pattern on the sweater. He will not place any patterns that leaves the placements of patterns asymmetric. Now he has to know how much empty space he should leave at the back of the sweater to achieve this.

The empty space that is not covered by the patterns must be a contiguous (possibly empty) section at the back of the sweater.

Image by Linn Bryhn Jacobsen, from commons.wikimedia.org

Input

The input contains two positive integers N, the length of the sweater, and P, the length of the pattern. They satisfy $1 \leq P \leq N \leq 10^{18}$ and they have the same parity, as otherwise the pattern could never be perfectly centered.

Output

Print a single integer, the amount of empty space left on the back of the sweater.

Explanation of samples

In the first sample the sweater is 13 loops in circumference. Thus the centered pattern is placed at loops 6, 7 and 8. There's space for another pattern in either direction at loops 3, 4, 5 and 9, 10, 11. There's not enough space to place two more, and a single pattern would make things asymmetric. Thus loops 1, 2, 12 and 13 are empty, so the answer is 4.

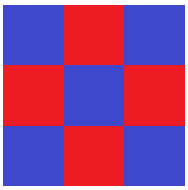


Figure K.1: A possible pattern for sample 1

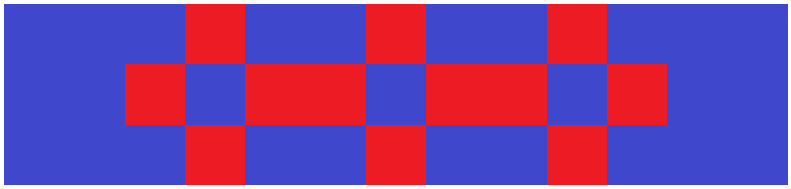


Figure K.2: Applying the pattern to the full width for sample 1

In the second sample the sweater is 16 loops in circumference. The first pattern is placed at loops 7, 8, 9 and 10. Two more are placed at 3, 4, 5, 6 and 11, 12, 13, 14. This leaves 1, 2, 15 and 16, which exactly fits one more pattern that will be perfectly centered at the back of the sweater, creating no asymmetry. Thus that pattern is placed, leaving no empty space.

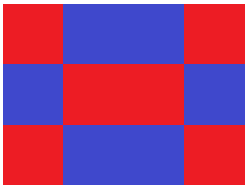


Figure K.3: A possible pattern for sample 2

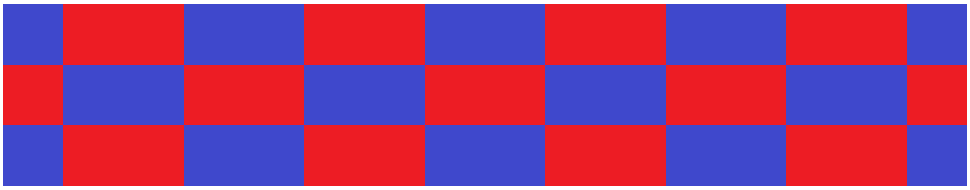


Figure K.4: Applying the pattern to the full width for sample 2

Sample Input 1	Sample Output 1
13 3	4

Sample Input 2	Sample Output 2
16 4	0