# 2018 Canadian Computing Olympiad
# Day 1, Problem 1
# **Geese vs. Hawks**

**Time Limit: 1 second**

**Problem Description**

Troy and JP are big hockey fans. Every hockey team played $N$ games this season. Each game was between two teams and the team that scored more points won. No game ended in a tie.

Troy's favourite team is the Waterloo Geese and he recorded the outcome of all their games as a string $S$. $S_i = \texttt{W}$ if the Geese won their $i$-th game; otherwise $S_i = \texttt{L}$ if the Geese lost their $i$-th game. He also recorded that they scored $A_i$ points in their $i$-th game.

JP's favourite team is the Laurier Hawks and he recorded the outcome of all their games as a string $T$. $T_j = \texttt{W}$ if the Hawks won their $j$-th game; otherwise $T_j = \texttt{L}$ if the Hawks lost their $j$-th game. He also recorded that they scored $B_j$ points in their $j$-th game.

Troy and JP recorded wins/losses and points in the order that their favourite teams played.

A *rivalry* game is one where the Geese and Hawks played each other. Since neither Troy or JP recorded the opponents their favourite teams faced, they are not sure which games, if any, were rivalry games. They wonder what is the maximum possible sum of points scored by both their teams in rivalry games that matches the information they recorded.

**Input Specification**

The first line contains one integer $N$ $(1 \leq N \leq 1\,000)$.

The second line contains string $S$ of length $N$ consisting of characters `W` and `L`.

The third line contains $N$ integers $A_1, \cdots, A_N$ $(1 \leq A_i \leq 1\,000\,000)$.

The fourth line contains string $T$ of length $N$ consisting of characters `W` and `L`.

The fifth line contains $N$ integers $B_1, \cdots, B_N$ $(1 \leq B_j \leq 1\,000\,000)$.

For 10 of the 25 available marks, $N \leq 10$.

**Output Specification**

Print one line with one integer, the maximum possible sum of points scored in potential rivalry games.

**Sample Input 1**
```
1
W
2
W
3
```

**Output for Sample Input 1**
```
0
```

**Explanation for Output for Sample Input 1**
Since both the Geese and Hawks won all their games, there could not have been any rivalry games.

**Sample Input 2**
```
4
WLLW
1 2 3 4
LWWL
6 5 3 2
```

**Output for Sample Input 2**
```
14
```

**Explanation of Output for Sample Input 2**
The fourth game each team played could have been a rivalry game where Geese won with $4$ points to the Hawk's $2$ points. The third game the Geese played and the second game the Hawks played could have been a rivalry game where the Hawks won with $5$ points compared to $3$ points of the Geese. The points scored by both teams is $4 + 2 + 5 + 3 = 14$ and this is the maximum possible.

Note that the first game played by the Geese was a win where they scored 1 goal: this game cannot be against the Hawks, since there is no game where the Hawks scored 0 goals. Similarly, the first game played by the Hawks cannot be used, since the Hawks lost and scored 6 goals, and the Geese never had a game where they scored at least 7 goals.

# 2018 Canadian Computing Olympiad
# Day 1, Problem 2
# **Wrong Answer**

**Time Limit: 1 second**

## Problem Description

Troy made the following problem (titled WA) for a programming contest:

> There is a game with $N$ levels numbered from 1 to $N$. There are two characters, both are initially at level 1. For $i < j$, it costs $A_{i,j}$ coins to move a character from level $i$ to level $j$. It is not allowed to move a character from level $i$ to level $j$ if $i > j$. To win the game, every level (except level 1) must be visited by exactly one character. What is the minimum number of coins needed to win?

JP is a contestant and submitted the following Python solution.

```
def Solve(N, A):
  # A[i][j] is cost of moving from level i to level j
  # N is the number of levels
  x, y, sx, sy = 1, 1, 0, 0 # Initialize x and y to 1, sx and sy to 0
  for i in range(2, N + 1): # loop from 2 to N
    if sx + A[x][i] < sy + A[y][i]:
      sx += A[x][i]
      x = i
    else:
      sy += A[y][i]
      y = i
  return sx + sy
```

Troy is certain that JP's solution is wrong. Suppose for an input to WA, JP's solution returns $X$ but the minimum number of coins needed is $Y$. To show how wrong JP's solution is, help Troy find an input $N$ and $A_{i,j}$ such that $\frac{X}{Y}$ is maximized.

## Input Specification

There is no input.

## Output Specification

Print an input to WA in the following format:

On the first line, print one integer $N$ $(2 \leq N \leq 100)$.
Then print $N-1$ lines; the $i$-th line should contain $N-i$ integers $A_{i,i+1}, \cdots, A_{i,N}$ $(1 \leq A_{i,j} \leq 100)$.

If your output is not the correct format, it will get an *incorrect* verdict on the sample test in the grader and score 0 points.

Otherwise, suppose that for your input, JP's solution returns $X$ but the minimum number of coins needed is $Y$. Then you will receive $\left\lceil \min(25, \frac{X}{4Y}) \right\rceil$ points where $\lceil Z \rceil$ is the smallest integer that is not greater than $Z$.

**Sample Output**

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

**Explanation for Sample Output**

The optimal way to win the game is for one character to visit level $2$ and the other character to visit levels $3$, $4$ and $5$. This costs $(1) + (2 + 7 + 5) = 15$ coins. JP's solution returns $18$. Thus $\frac{X}{4Y} = \frac{18}{4 \times 15} = 0.3$, so this output will receive $\lceil 0.3 \rceil = 1$ point.

# 2018 Canadian Computing Olympiad
# Day 1, Problem 3
# **Fun Palace**

**Time Limit: 1 second**

**Problem Description**

You are working hard to prepare a fun party for your fun friends. Fortunately, you have just located the perfect venue for the fun party: a *fun palace*. The fun palace has $N$ *fun rooms* connected in a linear structure. The fun rooms are numbered from 1 to $N$, and for $1 \leq i \leq N-1$, fun rooms $i$ and $i+1$ are connected by a *fun tunnel*. We say that such a fun tunnel is *incident* to fun rooms $i$ and $i+1$. In addition, fun room 1 is incident to an *exit tunnel* leaving the fun palace.

Fun tunnels can be in one of two states: open or closed. When the fun tunnel between fun rooms $i$ and $i+1$ is opened, fun friends may travel freely between the two rooms, in either direction.

By default, the fun tunnels will all be closed. However, they may temporarily be opened by a group of fun friends pressing down a required number of *fun buttons*. For each fun tunnel, there is a set of fun buttons present in the fun rooms connected to the fun tunnel. If all of the buttons in one of the rooms connected to a tunnel are pressed down by distinct fun friends, then the fun tunnel will open. Otherwise, the fun tunnel will immediately close. The fun tunnel between rooms $i$ and $i+1$ is connected to a set of $a_i$ buttons in room $i$ and a set of $b_i$ buttons in room $i+1$. To put this another way, if there are at least $a_i$ friends in room $i$ or if there are $b_i$ friends in room $i+1$, then tunnel between room $i$ and $i+1$ may be opened.

The exit tunnel operates under similar rules, but it is only connected to a single set of $e$ buttons present in room 1.

You want to ensure your friends have maximum fun, and that obviously means keeping your fun friends trapped in the fun palace forever. What is the maximum number of fun friends that you can distribute to particular fun rooms such that the exit fun tunnel is never opened?

**Input Specification**

The first line will contain a single integer $N$ ($1 \leq N \leq 1000$), the number of fun rooms. The next line contains a single integer $e$ ($1 \leq e \leq 10\,000$). The next $N-1$ lines contain two space-separated integers each, with the $i$th of these lines containing $a_i$ and $b_i$ ($1 \leq a_i, b_i \leq 10\,000$).

For 3 of the 25 marks available, $1 \leq e \leq 200$, $a_i = 1$, $b_i = 1$.

For an additional 5 of the 25 marks available, $1 \leq e, a_i, b_i \leq 2$.

For an additional 12 of the 25 marks available, $N \leq 200$, $1 \leq e, a_i, b_i \leq 200$.

**Output Specification**

Output a single integer, the maximum number of fun friends over all possible distributions of fun friends to fun rooms such that there is no way for the fun friends to open the exit tunnel.

**Sample Input 1**
```
2
20
5 5
```

**Output for Sample Input 1**
```
19
```

**Explanation for Output for Sample Input 1**
If we had any more than 19 fun friends, then they would be able to move to fun room 1 and press all 20 fun buttons required in order to open the exit tunnel.

**Sample Input 2**
```
2
20
5 20
```

**Output for Sample Input 2**
```
24
```

**Explanation for Output for Sample Input 2**
Suppose we place 24 fun friends in fun room 2. In order to open the fun tunnel between fun rooms 1 and 2, 20 fun friends must stay in fun room 2 to press the fun buttons. This allows only 4 fun friends into fun room 1, which is not enough to press every fun button in the set of 5.

**Sample Input 3**
```
7
7
2 8
6 6
1 1
2 4
2 8
7 8
```

**Output for Sample Input 3**
```
23
```

**Explanation for Output for Sample Input 3**
One optimum distribution is to place 9 fun friends in fun room 2 and 14 fun friends in fun room 7.

# 2018 Canadian Computing Olympiad
# Day 2, Problem 1
# **Gradient Descent**

**Time Limit: 1 second**

**Problem Description**
Troy wants to play the following game with you.

He has a grid with $R$ rows and $C$ columns. Rows are numbered from 1 to $R$ and columns are numbered from 1 to $C$. Let the cell at row $p$ and column $q$ be denoted as $(p, q)$.

There are $N$ tokens numbered from 1 to $N$. Token $i$ is placed at $(X_i, Y_i)$ where $1 \leq X_i \leq R$ and $1 \leq Y_i \leq C$. There may be multiple tokens at the same cell. In one second, Troy can move one token to a horizontally or vertically adjacent cell. The *score* of a cell is defined as the minimum number of seconds needed for Troy to move every token to this cell.

Your goal is to find the minimum score of any cell in the grid. Unfortunately, Troy does not tell you how many tokens there are or where they are placed. However, you may ask him questions. You can ask Troy to tell you the score of any cell $(p, q)$. You may ask at most $K$ questions before Troy gets bored.

**Interaction Protocol**
This problem is interactive: input will be given based on questions generated by your program.

First, read one line with three integers $R$, $C$, $K$ $(1 \leq R, C \leq 10\,000\,000; 1 \leq K \leq 170)$.

After your program has read this line, your program may ask questions.

To ask a question about $(p, q)$ $(1 \leq p \leq R; 1 \leq q \leq C)$, print one line in the format "? $p$ $q$". Then, read one line with one integer $s$ $(0 \leq s \leq 2\,000\,000\,000)$, the score of $(p, q)$.

Once your program determines the minimum score is $Z$, print one line in the format "! $Z$". Your program must terminate immediately after printing this line.

The output must be flushed after every line is printed, including the last line. To flush you can use: `fflush(stdout)` or `cout << endl` in C/C++; `System.out.flush()` in Java; `flush(output)` in Pascal.

If any printed line is wrongly formatted or you ask more than $K$ questions you will get an *incorrect* verdict.

For every test case, the grading system will have fixed integer values $N$, $R$, $C$, $K$, $X_1$, ..., $X_N$, $Y_1$, ..., $Y_N$ $(1 \leq N \leq 100; 1 \leq X_i \leq R; 1 \leq Y_i \leq C)$. These values will remain constant while your program is running. That is, the grading system is **not adaptive**.

For $5$ of the $25$ available marks, $R = 1$, $C \leq 90$ and $K = 90$.

For an additional $5$ of the $25$ available marks, $R = 1$, and $K = 90$.

For an additional $5$ of the $25$ available marks, $K = 170$.

For an additional $5$ of the $25$ available marks, $K = 100$.

For the remaining $5$ marks, $K = 75$.

**Sample Interaction 1**

| Request to grader | Feedback from Grader |
|---|---|
| | `1 10 90` |
| `? 1 3` | `9` |
| `? 1 7` | `11` |
| `? 1 4` | `8` |
| `! 8` | |

**Explanation of Output for Sample Input 1**
This sample corresponds to tokens at cells $(1, 2)$, $(1, 4)$ and $(1, 10)$. It is guaranteed this sample will match the sample test in the grader.

The score of cell $(1, 3)$ is $1 + 1 + 7 = 9$.

The score of cell $(1, 7)$ is $5 + 3 + 3 = 11$.

The score of cell $(1, 4)$ is $2 + 0 + 6 = 8$ and this is the minimum in the grid.

For information, here are the scores in this example:

| Column | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Score | 13 | 10 | 9 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**Sample Interaction 2**

| Request to grader | Feedback from Grader |
|---|---|
| | `5 4 170` |
| `? 2 4` | `11` |
| `? 1 4` | `15` |
| `? 3 3` | `7` |
| `! 7` | |

**Explanation of Output for Sample Input 2**
This sample corresponds to tokens at cells $(2, 3)$, $(2, 3)$, $(4, 3)$ and $(5, 1)$.

# 2018 Canadian Computing Olympiad
# Day 2, Problem 2
# **Boring Lectures**

**Time Limit: 8 seconds**

**Problem Description**

You have a schedule of $N$ upcoming lectures that you have the option of attending. The lectures are numbered from 1 to $N$ and are in chronological order. From the current schedule, you expect that the $i$th lecture will have quality $a_i$. Since most of the lectures will be boring, you are only willing to attend some group of $K$ consecutive lectures. You will skip the remaining lectures so that you can catch up on sleep and participate in programming contests. Since you don't like taking notes, you will only be able to remember the content from 2 of the lectures you attend. You want to choose the lectures you attend and the 2 lectures you remember as to maximize the sum of the lecture qualities of those 2 lectures.

There are $Q$ changes that will be made to the schedule. The $j$th change to the schedule is represented by two values $i_j, x_j$ that indicate that the quality of the $i_j$th lecture changes to $x_j$. For each of the $Q + 1$ versions of the schedule, find the maximum possible sum of lecture qualities that you can attain.

**Input Specification**

The first line will contain three integers $N$, $K$, and $Q$ ($2 \leq N \leq 10^6$, $2 \leq K \leq N$, $0 \leq Q \leq 10^5$). The second line contains $N$ integers $a_1, \ldots, a_N$, $(0 \leq a_i \leq 10^9)$. The next $Q$ lines each contain two integers $i_j$ and $x_j$ ($1 \leq i_j \leq N$, $0 \leq x_j \leq 10^9$).

For 5 of the 25 available marks, $Q = 0$.

For an additional 10 of the 25 available marks, $N \leq 10^4$.

**Output Specification**

Output $Q + 1$ lines, each containing a single integer. The $j$-th line that follows should contain the answer for the schedule obtained after the first $j - 1$ changes are made.

**Sample Input**

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

**Output for Sample Input**

```
8
6
```

**Explanation for Output for Sample Input**
For the original schedule, it is best to attend the first three lectures and remember the first and third, for an overall value of $6 + 2 = 8$. After the update, it is best to attend the last three lectures and remember the last two, giving a value of $2 + 4 = 6$.

# 2018 Canadian Computing Olympiad
# Day 2, Problem 3
# **Flop Sorting**

**Time Limit: 2 seconds**

## Problem Description

Desperate to contribute to the CCO, Robert tried inventing a segment tree problem. The specification for that problem is:

> You are given a list of $N$ distinct integers between $1$ and $N$. These are arranged in a row such that the $i$-th integer from the left is $a_i$, where $1 \leq i \leq N$. We define a *flop* operation on a set of elements as swapping the minimum element of the set with the maximum element of the set. There will be $Q$ flop operations, each specifying two numbers $l$ and $r$ ($1 \leq l \leq r \leq N$). For each operation, you must perform a *flop* on the interval $[l, r]$ (that is, on the segment of numbers $a_l, a_{l+1}, \ldots, a_{r-1}, a_r$). You must perform the $Q$ flops in the order given, and report the final result.

Now that he is finished with the problem statement, Robert needs to create some test data. For one test case in particular, he is trying to encode an inside joke into the initial and final sequences of numbers. With these two sequences fixed, help him find any sequence of flop operations that transforms the first sequence into the second.

## Input Specification

The first line will contain the integer $N$ ($1 \leq N \leq 4096$). The second line will contain $N$ distinct space separated integers between $1$ and $N$, representing the initial sequence. The third line will also contain $N$ distinct space separated integers between $1$ and $N$, representing the final sequence.

## Output Specification

The first line of output should contain the integer $Q$, with $Q \leq 300\ 000$. The next $Q$ lines should contain two integers $l$ and $r$ with $1 \leq l \leq r \leq N$.

For 5 of the available 25 marks, $N \leq 100$.

For an additional 10 of the available 25 marks, $N \leq 2048$.

## Sample Input

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

## Output for Sample Input

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

**Explanation for Output for Sample Input**
The first flop swaps 3 and 5, the second flop swaps 6 and 2, the third flop swaps 5 and 2, and the fourth flop swaps 5 and 4.