08. Mapping 1

Permutation
Mapping (not necessarily one-to-one)
Sorting and searching (“`indexOf()`” in ArrayList)

Exercise 1: Times table

Exercise 2: Finding primes

Exercise 3: Max after swap

Exercise 4: Shuffle (Dec 2017)

Exercise 5: Hoof, paper, scissors (Jan 2017)

Exercise 6: Blocks (Feb 2022)

Times table

Perform a mapping: 0 → A, 1→ B, … 9 → J, display the times table using letters A~J.

Perform a mapping: 0 → 1, 1→ 2, … 9 → 0, display the times table.

Original table:

```
  1  2  3  4  5  6  7  8  9
  2  4  6  8 10 12 14 16 18
  3  6  9 12 15 18 21 24 27
  4  8 12 16 20 24 28 32 36
  5 10 15 20 25 30 35 40 45
  6 12 18 24 30 36 42 48 54
  7 14 21 28 35 42 49 56 63
  8 16 24 32 40 48 56 64 72
  9 18 27 36 45 54 63 72 81
```

Under mapping 1:

```
  B  C  D  E  F  G  H  I  J
  C  E  G  I BA BC BE BG BI
  D  G  J BC BF BI CB CE CH
  E  I BC BG CA CE CI DC DG
  F BA BF CA CF DA DF EA EF
  G BC BI CE DA DG EC EI FE
  H BE CB CI DF EC EJ FG GD
  I BG CE DC EA EI FG GE HC
  J BI CH DG EF FE GD HC IB
```

Under mapping 2:

```
  2  3  4  5  6  7  8  9  0
  3  5  7  9 21 23 25 27 29
  4  7  0 23 26 29 32 35 38
  5  9 23 27 31 35 39 43 47
  6 21 26 31 36 41 46 51 56
  7 23 29 35 41 47 53 59 65
  8 25 32 39 46 53 50 67 74
  9 27 35 43 51 59 67 75 83
  0 29 38 47 56 65 74 83 92
```

Finding primes

Given positive integers $n_1 < n_2$, find the count of prime numbers between n_1 and n_2, inclusive.

Input (from terminal / stdin)
- The only contains integers n_1 and n_2, $1 < n_1 < n_2 \leq 1e6$.

Output (to terminal / stdout)
- Report the prime numbers in $[n_1, n_2]$.

Sample input
2 10

Sample output
4

Max after swap

Given a very large positive integer, you could swap two digits at most once to get the maximum valued number. Report the maximum valued number you could get. Note the integer will be represented as a string.

Input (from terminal / stdin)

- The only line contains the integer, with at most 100 digits.

Output (to terminal / stdout)

- Report the max value after at most one swap of two digits.

Sample input
1993

Sample output
9913

Shuffle (Dec 2017)

Convinced that happy cows generate more milk, Farmer John has installed a giant disco ball in his barn and plans to teach his cows to dance!

Looking up popular cow dances, Farmer John decides to teach his cows the "Bovine Shuffle". The Bovine Shuffle consists of his N cows ($1 \leq N \leq 100$) lining up in a row in some order, then performing three "shuffles" in a row, after which they will be lined up in some possibly different order. To make it easier for his cows to locate themselves, Farmer John marks the locations for his line of cows with positions $1 \dots N$, so the first cow in the lineup will be in position 1, the next in position 2, and so on, up to position N.

A shuffle is described with N numbers, $a_1 \dots a_N$, where the cow in position j moves to position a_j during the shuffle (and so, each a_j is in the range $1 \dots N$). Every cow moves to its new location during the shuffle. Fortunately, all the a_j's are distinct, so no two cows try to move to the same position during a shuffle.

Farmer John's cows are each assigned distinct 7-digit integer ID numbers. If you are given the ordering of the cows after three shuffles, please determine their initial order.

INPUT FORMAT (file shuffle.in):

- The first line of input contains N, the number of cows.
- The next line contains the N integers $a_1 \dots a_N$.
- The final line contains the order of the N cows after three shuffles, with each cow specified by its ID number.

OUTPUT FORMAT (file shuffle.out):

- You should write N lines of output, with a single cow ID per line, specifying the order of the cows before the three shuffles.

SAMPLE INPUT:
```
5
1 3 4 5 2
1234567 2222222 3333333 4444444 5555555
```

SAMPLE OUTPUT:
```
1234567
5555555
2222222
3333333
4444444
```

Hoof, paper, scissors (Jan 2017)

You have probably heard of the game "Rock, Paper, Scissors". The cows like to play a similar game they call "Hoof, Paper, Scissors".

The rules of "Hoof, Paper, Scissors" are simple. Two cows play against each-other. They both count to three and then each simultaneously makes a gesture that represents either a hoof, a piece of paper, or a pair of scissors. Hoof beats scissors (since a hoof can smash a pair of scissors), scissors beat paper (since scissors can cut paper), and paper beats hoof (since the hoof can get a papercut). For example, if the first cow makes a "hoof" gesture and the second a "paper" gesture, then the second cow wins. Of course, it is also possible to tie, if both cows make the same gesture.

Farmer John watches in fascination as two of his cows play a series of N games of "Hoof, Paper, Scissors" ($1 \leq N \leq 100$). Unfortunately, while he can see that the cows are making three distinct types of gestures, he can't tell which one represents "hoof", which one represents "paper" and which one represents "scissors" (to Farmer John's untrained eye, they all seem to be variations on "hoof"...)

Not knowing the meaning of the three gestures, Farmer John assigns them numbers 1, 2, and 3. Perhaps gesture 1 stands for "hoof", or maybe it stands for "paper"; the meaning is not clear to him. Given the gestures made by both cows over all N games, please help Farmer John determine the maximum possible number of games the first cow could have possibly won, given an appropriate mapping between numbers and their respective gestures.

INPUT FORMAT (file hps.in):

- The first line of the input file contains N.
- Each of the remaining N lines contain two integers (each 1, 2, or 3), describing a game from Farmer John's perspective.

OUTPUT FORMAT (file hps.out):

- Print the maximum number of games the first of the two cows could possibly have won.

SAMPLE INPUT:

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

SAMPLE OUTPUT:

```
2
```

One solution (of several) for this sample case is to have 1 represent "scissors", 2 represent "hoof", and 3 represent "paper". This assignment gives 2 victories to the first cow ("1 3" and "3 2"). No other assignment leads to more victories.

Blocks (Feb 2022)

In an effort to improve her vocabulary, Bessie the cow has obtained a set of four wooden blocks, each one a cube with a letter of the alphabet written on each of its six sides. She is learning how to spell by arranging the blocks in a row so the letters on top of the blocks spell words.

Given the letters on each of Bessie's four blocks, and a list of words she would like to spell, please determine which of words on her list she will be able to spell successfully using the blocks.

INPUT FORMAT (input arrives from the terminal / stdin):

- The first line of input contains N $(1 \leq N \leq 10)$, the number of words that Bessie would like to spell.
- The next four lines each contain a string with six uppercase letters, representing the letters on the six sides of one of Bessie's blocks.
- The next N lines contain the N words Bessie would like to spell. Each of these is between 1 and 4 uppercase letters long.

OUTPUT FORMAT (print output to the terminal / stdout):

- For each word on Bessie's list, output YES if she is able to spell it using the blocks and NO otherwise.

SAMPLE INPUT:
```
6
MOOOOO
OOOOOO
ABCDEF
UVWXYZ
COW
MOO
ZOO
MOVE
CODE
FARM
```

SAMPLE OUTPUT:
```
YES
NO
YES
YES
NO
NO
```

In this example, Bessie can spell COW, ZOO, and MOVE. Sadly, she cannot spell MOO, since the only block with an M cannot also be used for an O. She cannot spell FARM since there is no block with a letter R. She cannot spell CODE since the C, D, and E all belong to the same block.