C_

Task 1 | Task 2

Challenge 1 description

A string made of an **even** number of characters ("<" and/or ">") is called symmetric if all characters in its first half are "<" and all characters in its second half are ">". Examples of symmetric strings are: "" (empty string), "<>", "<<>>", "<<<>>>", etc.

Write a function:

```
int solution(char *S);
```

that, given a string S made of N characters ("<", ">" and/or "?")", returns the length of the longest symmetric substring that can be obtained after replacing question marks with "<" or ">" characters.

Examples:

1. For S = "<><??>>", after replacing all question marks with "<", the string "<><<<>>" is obtained. Its longest symmetric substring is "<<>>", so the function should return 4.

2. For S = "??????", the optimal option is to replace the first three question marks with "<" and the next three question marks with ">", thus obtaining the string "<<<>>>". The function should return 6.

3. For S = "<<?", the function should return 2.

Write an **efficient** algorithm for the following assumptions:

- the length of string S is within the range [1..200,000];
- string S is made only of the following characters: '<', '>' and/or '?'.

Solution C

```
1  // you can write to stdout for debugging purposes, e.g.
2  // printf("this is a debug message\n");
3
4  int solution(int **A, int N, int M) {
5      // write your code in [LANGVER]
6  }
7
```

Test output

C_ Task 1 Task 2 Time left: 1 h 59 min

Challenge 2 description

There is a list A made of N strings, each of length 2. Two strings can be connected together if the letter ending the first string is the same as the letter beginning the second string. For example, the two strings "ab" and "bc" can be connected, as "ab" ends with a "b" character and "ba" begins with a "b" character. Potentially more strings can be connected to form a longer chain if the last letter of the previous string is the same as the first letter of the next string, e.g. "ab"-"ba"-"ac"-"cb".

For every K from 0 to N−1, check if all the strings A[0], ..., A[K] can be connected together. **Strings can be reordered**, but must not be reversed.

Write a function:

```
char * solution(char *A[], int N);
```

that, given an array A consisting of N strings, each of length 2, returns a string of length N. The K-th character of the string should be '1' if strings A[0], ..., A[K] can be connected into a single string, or '0' otherwise.

Examples:

1. For A = ["he", "ll", "lo", "el"], the function should return "1001":

- ["he"] is already a single string → 1
- ["he", "ll"] cannot be connected → 0
- ["he", "ll", "lo"] cannot be connected → 0
- ["he", "ll", "lo", "el"] can be connected into "he"-"el"-"ll"-"lo" → 1

2. For A = ["ab", "ba", "bq"], the function should return "111". Strings can be connected into a single string for every K = 0, 1 and 2. Note that ["ab", "ba"] can be connected into

Solution C

```
// you can write to stdout for debugging purposes, e.g.
// printf("this is a debug message\n");

char * solution(char *A[], int N) {
    // Implement your solution here
}
```

Test output

C_ Task 1 Task 2 Time left: 1 h 59 min

Write a function:

```
char * solution(char *A[], int N);
```

that, given an array A consisting of N strings, each of length 2, returns a string of length N. The K-th character of the string should be '1' if strings A[0], ..., A[K] can be connected into a single string, or '0' otherwise.

Examples:

1. For A = ["he", "ll", "lo", "el"], the function should return "1001":

- ["he"] is already a single string → 1
- ["he", "ll"] cannot be connected → 0
- ["he", "ll", "lo"] cannot be connected → 0
- ["he", "ll", "lo", "el"] can be connected into "he"-"el"-"ll"-"lo" → 1

2. For A = ["ab", "ba", "bq"], the function should return "111". Strings can be connected into a single string for every K = 0, 1 and 2. Note that ["ab", "ba"] can be connected into either "ab"-"ba" or "ba"-"ab".

3. For A = ["ee", "ea", "eg"], the function should return "110".

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- each string in A has length 2 and consists only of lowercase letters (a-z).

Solution C

```c
// you can write to stdout for debugging purposes, e.g.
// printf("this is a debug message\n");

char * solution(char *A[], int N) {
    // Implement your solution here
}
```

Test output