02

Resolvendo o problema

Transcrição

Continuaremos a resolução do problema do palíndromo, precisamos justamente descobrir se a palavra que o programa lê é um palíndromo.

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.println(i + " ");
            String palavra = scanner.nextLine();

        }
    }
}
```

Para verificar se a palavra é um palíndromo ou não, temos que ler caractere a caractere da esquerda para a direita e ver se eles também aparecem no final da palavra. Para isso, usaremos um `for()`, que irá do início da `palavra` (`i = 0`), até o fim da `palavra`, ou seja: `palavra.lenght()`. Encaixando isso em nosso código:

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.println(i + " ");
            String palavra = scanner.nextLine();
            for(int i = 0; i < palavra.lenght(); i++){

            }

        }
    }
}
```

Agora é preciso comparar se o caractere que está na posição inicial é o mesmo que está na posição final. Temos que parear o caractere na posição `0` com o último, o primeiro com o último menos 1, e assim por diante. Assim, temos que diminuir `i` de `palavra.lenght`. Ou algo parecido com isso:

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
```

```
            System.out.println(i + " ");
            String palavra = scanner.nextLine();
            for(int i = 0; i < palavra.lenght(); i++){
              if(palavra.charAt(i)== palavra.charAt(palavra.lenght() - 1){

              }

            }

        }
    }
}
```

Se isso for verdade, será um palíndromo. Se não for (`else`), não é palíndromo. Precisamos ainda adicionar um `boolean` . Repare que o código está bem feio.

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.println(i + " ");
            String palavra = scanner.nextLine();

            boolean palindromo = true;
            for(int i = 0; i < palavra.lenght(); i++){
              if(palavra.charAt(i)== palavra.charAt(palavra.lenght() - 1)
                  //uhu!
              } else {
                palindromo = false;

              }

            }
            System.out.println(palindromo);
        }
    }
}
```

Adicionei ao final um `sysout` para que possamos testar esse novo trecho de código. Mesmo que ele não imprima o `"YES"` ou `"NO"` , mas `true` e `false` . Indo para o terminal, temos:

```
Alura-Azul:bin alura$ java Main < entrada1.txt
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
          Duplicate local variable i

          at Main.main(Main.java:12)
Alura-Azul:bin alura$
```

Deu erro de compilação. Lembra que mencionei que temos que tomar muito cuidado ao escrever o `for()` e que esse trecho do código estava feio? Pelo menos é um erro de compilação; declaramos uma variável `i` , sendo que ela já existe.

Mesmo em maratonas de programação, que é um lugar no qual há a visão preconceituosa de que é preciso digitar o mais rápido possível, e tudo bem se ficar feio. Mas não é assim, é preciso tomar cuidado com o seu código. Senão você usa uma variável que já existe. Por exemplo, podemos colocar apenas `i = 0` na segunda ocorrência.

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.println(i + " ");
            String palavra = scanner.nextLine();

            boolean palindromo = true;
            for(i = 0; i < palavra.lenght(); i++){
              if(palavra.charAt(i)== palavra.charAt(palavra.lenght() - 1)
                  //uhu!
              } else {
                palindromo = false;

              }

            }
            System.out.println(palindromo);
        }
    }
}
```

Agora testaremos no terminal.

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1
true1
Exception in thread "main" java.lang.StringIndexOutOfBoundExeption: String index out of range: 
      at java.lang.String.charAt(String.java:646)
      at Main.main(Main.java:13)
```

Deu outro erro esquisito, e o que ele imprimiu também está bem estranho. A primeira string realmente era um palíndromo, então o `true` está ok. Mas depois ele imprimiu o número `1` de novo. Ele deveria imprimir o número `2`. Antes estava tudo certo, o que eu fiz?!

Alguns programadores sugeririam que declarássemos todas as variáveis no começo, um resquício de algumas linguagens que já não precisam disso. Mas mesmo se fizéssemos essa declaração o erro seria o mesmo. E o resultado continuaria não fazendo sentido nenhum. Esse é um exemplo de péssima prática de programação, porque podemos criar variáveis que não ser utilizadas, e vamos criar variáveis que serão reutilizadas em outro contexto – o que bagunça tudo. Também temos que considerar que depois desse código essa variáveis continuam acessíveis, então teremos que pensar com muito cuidado em seus valores, e mesmo assim é capaz que você use pensando que está com um valor quando na verdade está com outro.

Então vamos nos livrar dessa prática de declarar variáveis no início, que era obrigatória em uma linguagem no passado, por causa da maneira que ela foi feita, no padrão que ela existia. Se hoje em dia, no padrão de outra linguagens você não precisa disso, não faça.

Por isso, vamos declarar uma variável nova todas as vezes. E o nosso compilador, que é nosso amigo, vai nos avisar que a variável já existe. Vamos cortar esse código bagunçado e recomeçar esse trecho.

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.println(i + " ");
            String palavra = scanner.nextLine();

        }
    }
}
```

Ainda precisamos verificar se `palavra` é palíndromo ou não. Vamos colocar direto um `if()` logo depois da sua string, já pedindo para que imprima `"YES"` se for palíndromo.

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.println(i + " ");
            String palavra = scanner.nextLine();
            if(ehPalindromo(palavra)) {
              System.out.println("YES");
            }

        }
    }
}
```

Caso contrário (`else`), precisamos imprimir `"NO"` .

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.println(i + " ");
            String palavra = scanner.nextLine();
            if(ehPalindromo(palavra)) {
              System.out.println("YES");
            } else {
              System.out.println("NO");
            }

        }
```

```
    }
}
```

Ainda precisamos fazer a função `ehPalindromo`, e vamos fazer de uma maneira muito simples: `return false`.

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.println(i + " ");
            String palavra = scanner.nextLine();
            if(ehPalindromo(palavra)) {
              System.out.println("YES");
            } else {
              System.out.println("NO");
            }

        }
    }

    private static boolean ehPalindromo(String palavra) {
      return false;
    }
}
```

E isso é só para que possamos testar se a saída está certa ou errada. Não custa nada fazer o teste, demora aproximadamente dois segundos.

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1
NO
2
NO
```

E em dois segundos descobrimos que tem algo errado. Os números e as palavras estão saindo em linhas diferentes. Por isso temos que testar rápido e falhar rápido. Muito melhor do que só testar depois de escrever um monte de código.

Quando estamos quebrando a linha? Em `System.out.prinln(i + " ");`, logo depois do espaço. Vamos resolver isso tirando o `ln`:

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.print(i + " ");
            String palavra = scanner.nextLine();
            if(ehPalindromo(palavra)) {
              System.out.println("YES");
```

```
            } else {
              System.out.println("NO");
            }

        }
    }

    private static boolean ehPalindromo(String palavra) {
      return false;
    }
}
```

Vamos testar de novo?

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1
NO
2
NO
Alura-Azul:bin alura$ java Main < entrada1.txt
1 NO
2 NO
```

Agora sim. Vamos testar o `"YES"` , pedindo para que a função `ehPalindromo` dê um `return true` .

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.print(i + " ");
            String palavra = scanner.nextLine();
            if(ehPalindromo(palavra)) {
              System.out.println("YES");
            } else {
              System.out.println("NO");
            }

        }
    }

    private static boolean ehPalindromo(String palavra) {
      return true;
    }
}
```

Voltando para o terminal:

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1
NO
2
```

```
NO
Alura-Azul:bin alura$ java Main < entrada1.txt
1 NO
2 NO
Alura-Azul:bin alura$ java Main < entrada1.txt
1 YES
2 YES
```

Também funciona. Mas faltou um detalhe: as aspas, que o cliente pede na especificação. Vamos inseri-las?

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.print(i + " ");
            String palavra = scanner.nextLine();
            if(ehPalindromo(palavra)) {
              System.out.println("\"YES\"");
            } else {
              System.out.println("\"NO\"");
            }

        }
    }

    private static boolean ehPalindromo(String palavra) {
      return true;
    }
}
```

Novamente, vamos testar no terminal!

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1 "YES"
2 "YES"
```

O `"YES"` está ok. E o `"NO"` ?

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.print(i + " ");
            String palavra = scanner.nextLine();
            if(ehPalindromo(palavra)) {
              System.out.println("\"YES\"");
            } else {
              System.out.println("\"NO\"");
            }
```

```
        }
    }

    private static boolean ehPalindromo(String palavra) {
      return false;
    }
}
```

Voltando ao terminal:

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1 "YES"
2 "YES"
Alura-Azul:bin alura$ java Main < entrada1.txt
1 "NO"
2 "NO"
```

Vê como mudanças pequenas são rápidas de testar? São também rápidas de corrigir. Quando escrevemos um algoritmo complexo... Boa sorte! Eu mesmo estava perdido no código grande que tínhamos escrito para resolver o problema. E tinha erro mesmo no `lenght`, além do problema de uso indevido da variável `i`, mas fica difícil de ver sem testar pedaço a pedaço. Mudou uma vírgula, testou.

Com esses testes concluímos que a saída está ok. Então a leitura e a saída estão ok? Não. Não testamos a entrada ainda, então o faremos agora. Vamos direto para a função `ehPalindromo`. Por desencargo de consciência, vamos pedir para que imprima a `palavra`

```
    private static boolean ehPalindromo(String palavra) {
      System.out.println(palavra);
      return false;
    }
```

Vamos testar no terminal:

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1
"NO"
2 HELloollEH
"NO"
```

Nossa entrada não está ok. A primeira palavra não é vazia, e a segunda não é `HELloollEH`. Até agora estávamos lendo a entrada errado! Ele deveria ler `HELloollEH` e `ILOVEyou`. Veja a importância de testar cada vírgula. E não tem problema testar infinitas vezes. Quando foi que pulamos o teste?

Foi na linha `String palavra = scanner.nextline()`. Parecia bobo testar essa linha, então passamos direto para o `if()`. Mas precisávamos ter testado a leitura da entrada. O que será que está acontecendo para o programa ler uma linha em branco? Vamos ver nossa `entrada1.txt` novamente:

```
2
HELloollEH
```

```
ILOVEyou
```

Não tem uma linha em branco antes da primeira palavra. O que ele fez antes de ler o `HElloollEH` ?

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        for(int i = 1; i < casos+1; i++){
            System.out.print(i + " ");
            String palavra = scanner.nextLine();
            if(ehPalindromo(palavra)) {
              System.out.println("\"YES\"");
            } else {
              System.out.println("\"NO\"");
            }

        }
    }

    private static boolean ehPalindromo(String palavra) {
      return false;
    }
}
```

Ele leu um `nextInt` antes. E tem uma linha em branco depois dele? Sim! Se não conhecermos direito a API do `Scanner` , não saberemos que o `nextInt` lê o número, mas ignora os espaços em branco depois do número, inclusive a quebra de linha. Ela ainda precisa ser lida, e quem fez isso foi o `nextLine` . E não importa se eu conheço essa API muito bem, é preciso testar. Escreveu uma nova linha? Testa. Escreveu mais uma? Testa de novo.

Em programas de maratona, nos quais estamos treinando nossa capacidade de entender e solucionar problemas, é muito fácil de ir testando no terminal. É um feedback muito rápido. Em problemas com estruturas mais complexas, com sistemas de bancos de dados, podemos criar testes unitários e usá-los a rodo. Aqui na plataforma temos cursos de testes unitários

Bom, já sabemos que estamos com um problema no `nextInt` e no `nextLine` . Temos várias maneiras de resolver isso. Poderíamos ler a linha inteira do número com o `nextLine` e depois transformar em `int` . Ou podemos acrescentar um `scanner.nextLine` , que é o que faremos.

```
public class Main {

    public static void main (String [] args) {
        Scanner scanner = new Scanner(System.in);
        int casos = scanner.nextInt();
        scanner.nextLine();

        for(int i = 1; i < casos+1; i++){
            System.out.print(i + " ");
            String palavra = scanner.nextLine();
            if(ehPalindromo(palavra)) {
              System.out.println("\"YES\"");
            } else {
```

```
                System.out.println("\"NO\"");
              }

          }
      }

      private static boolean ehPalindromo(String palavra) {
        return false;
      }
  }
```

Assim, ele vai ler até o fim da linha e ignorar. Escrevemos mais uma linha de código, então vamos testar.

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1 HELloollEH
"NO"
2 ILOVEyou
"NO"
```

Agora o programa leu corretamente. Como testamos essa linha e deu certo, podemos seguir, pois temos certeza de que `palavra` está certa. E devolvendo `true` ou `false`, o resultado fará sentido. Só falta escrever o código da função.

Nós separamos em pedaços, e é isso que costumo fazer: primeiro testar a entrada, depois a saída e só depois o algoritmo. Tenha certeza de que está lendo certo todos os casos possíveis, depois tenha certeza de que a saída está vindo do jeito certo – afinal, não adianta o algoritmo estar certinho se na hora de mostrar o resultado tudo muda. Depois, tenha certeza de que o algoritmo está certo. As três partes são importantes, mas lide com uma de cada vez.

Nós ainda estamos testando um caso só na `entrada1.txt`. Ainda não extrapolamos o que o cliente pode nos enviar, e nós o faremos antes de enviar para o servidor. Para isso, pensaremos nas formas que um juiz pode testar as nossas bordas, o que eles realmente fazem. Um cliente pode ter esquecido uma situação, e é nossa função apontar esse tipo de caso para ele e ajudá-lo a desenvolver essa capacidade crítica.

Podemos primeiro implementar o `ehPalindromo`. Já temos a leitura da palavra, que precisa começar do `0` até o fim da palavra. Se pensarmos na palavra `HELloollEH`, podemos escrever a linha:

```
for(int i = 0; i < palavra.length(); i++){

}
```

Mas qual é o lenght da palavra? Ao contarmos, vemos que são `10` caracteres. Mas será que nossa entrada não tem nenhum caractere em branco que ele vai contar? Precisamos ter certeza. Para isso, vamos pedir para o programa imprimir o `palavra.lenght`. Assim:

```
    private static boolean ehPalindromo(String palavra) {
      System.out.println(palavra + " " + palavra.lenght());
      for(int i = 0; i < palavra.length(); i++){

      }
      return false;
```

Rodando no terminal:

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1 HELloollEH 10
"NO"
2 ILOVEyou 8
"NO"
```

Agora sabemos que estávamos imaginando o caso certinho. Mas é bom saber se o programa está ou não lendo os espaços em branco. Se acrescentarmos um espaço depois da primeira palavra em `entrada1.txt` e rodarmos no terminal, teremos:

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1 HELloollEH  11
"NO"
2 ILOVEyou 8
"NO"
```

O programa está considerando o espaço em branco. Vê a sutileza da questão? E ainda nem saímos do exemplo dado no enunciado. Sabemos que o `i` começa em `0`. Ilustremos essa questão. A princípio, ele corresponderá ao primeiro `H`, e deverá ser pareado com o último `H`, indicado com um `*`:

```
//HELloollEH
//i        *
```

A última letra é a décima, mas como estamos contando como array, corresponderá à posição `9`. Ou seja, `10 - 1`. Para escrever isso, vamos explicitar que queremos comparar um caractere à esquerda com um à direita, identificando-os. Não há necessidade de escrever tudo na mesma linha, e mudar de linha nos ajuda a organizar o código.

```
private static boolean ehPalindromo(String palavra) {
  System.out.println(palavra + " " + palavra.lenght());
  for(int i = 0; i < palavra.length(); i++){
    char esquerda = palavra.charAt(i);
    char direita = palavra.charAt(palavra.lenght() - 1);

  }
  return false;
```

Dessa maneira, faz sentido comparar o caractere `0` com o `9`. Mas, e quando o `i` for maior? Por exemplo, quando `i` for `1`, vamos querer que ele seja comparado com a posição `8`. Ou seja, `10 - 1 -1`. E quando for `2`? Será a posição `7`, ou seja `10 - 1 -2`. Ou ainda: `10 - 1 -i`.

```
private static boolean ehPalindromo(String palavra) {
  System.out.println(palavra + " " + palavra.lenght());
  for(int i = 0; i < palavra.length(); i++){
    char esquerda = palavra.charAt(i);
    char direita = palavra.charAt(palavra.lenght() - 1 -i);
```

```
    }
    return false;
```

Assim, quando um lado andar mais um para a direita, o outro também o fará. Só falta comparar os dois. Se eles forem iguais, ficaremos felizes. Do contrário, `ehPalindromo = false` .

```
  private static boolean ehPalindromo(String palavra) {
    System.out.println(palavra + " " + palavra.lenght());
    for(int i = 0; i < palavra.length(); i++){
      char esquerda = palavra.charAt(i);
      char direita = palavra.charAt(palavra.lenght() - 1 -i);
      if(esquerda == direita){
        //feliz
      } else {
          ehPalindromo = false;
      }

    }
    return ehPalindromo;
```

Note que o `return` final agora é para o `ehPalindromo` . Acrescentaremos que, por padrão, `ehPalindromo` é verdadeiro no início desse bloco.

```
  private static boolean ehPalindromo(String palavra) {
    System.out.println(palavra + " " + palavra.lenght());

    boolean ehPalindromo = true;
    for(int i = 0; i < palavra.length(); i++){
      char esquerda = palavra.charAt(i);
      char direita = palavra.charAt(palavra.lenght() - 1 -i);
      if(esquerda == direita){
        //feliz
      } else {
          ehPalindromo = false;
      }

    }
    return ehPalindromo;
```

A função está fazendo o que queremos: verificar os caracteres um a um, inclusive se entrecruzando. Vamos testar?

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1 HELloollEH 10
"YES"
2 ILOVEyou 8
"NO"
```

Podemos ver passo a passo a conferência dos caracteres se cruzando, com outro `sysout` que mostre o `esquerda` e o `direita` :

```
  private static boolean ehPalindromo(String palavra) {
    System.out.println(palavra + " " + palavra.lenght());

    boolean ehPalindromo = true;
    for(int i = 0; i < palavra.length(); i++){
      char esquerda = palavra.charAt(i);
      char direita = palavra.charAt(palavra.lenght() - 1 -i);

      System.out.println(esquerda + " " + direita);
      if(esquerda == direita){
        //feliz
      } else {
          ehPalindromo = false;
      }

    }
    return ehPalindromo;
```

Ao salvar e imprimir no terminal, temos:

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1 HElloollEH 10
"YES"
2 ILOVEyou 8
"NO"
Alura-Azul:bin alura$ java Main < entrada1.txt
1 HElloollEH 10
H H
E E
L L
l l
o o
o o
l l
L L
E E
H H
"YES"
2 ILOVEyou 8
I u
L o
O y
V E
E V
y O
o L
u i
"NO"
```

Observe que passa de um sentido ao outro, cruzando as letras. Teoricamente, parece tudo ok com esse código. Vamos tirar os `sysout` s que colocamos para testar.

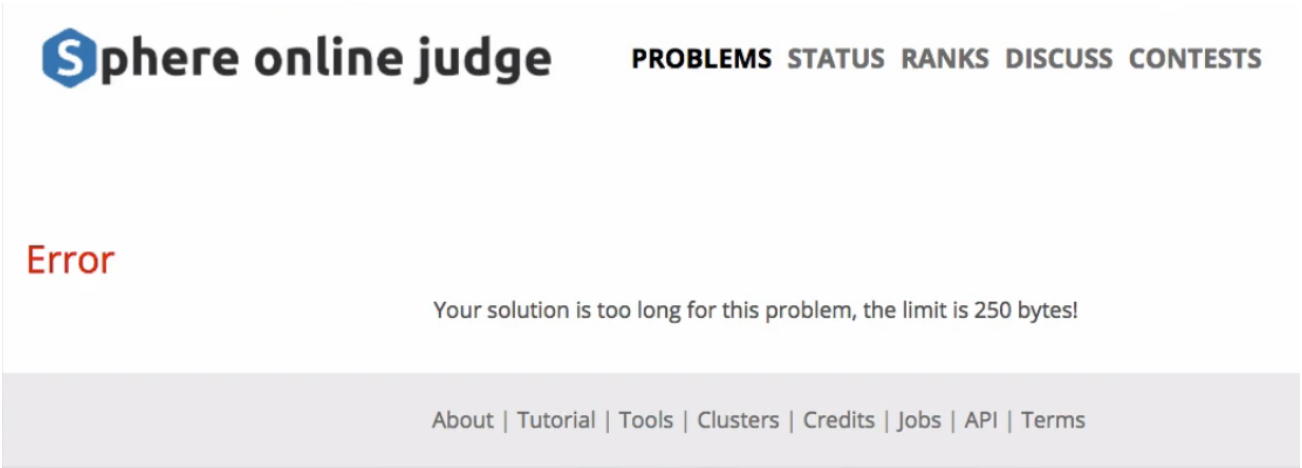
```
  private static boolean ehPalindromo(String palavra) {
    //System.out.println(palavra + " " + palavra.lenght());
```

```
    boolean ehPalindromo = true;
    for(int i = 0; i < palavra.length(); i++){
      char esquerda = palavra.charAt(i);
      char direita = palavra.charAt(palavra.lenght() - 1 -i);

      //System.out.println(esquerda + " " + direita);
      if(esquerda == direita){
        //feliz
      } else {
          ehPalindromo = false;
      }

    }
    return ehPalindromo;
```

Rodaremos de novo, por desencargo de consciência, já que fizemos alterações.

```
Alura-Azul:bin alura$ java Main < entrada1.txt
1 "YES"
2 "NO"
```

Parece estar funcionando. Podemos, então, submeter a solução no Spoj. É preciso avisar que estamos escrevendo em Java antes de colar o código no campo de submissão. Ao enviarmos, uma surpresa:



Deu erro! O site nos diz que a nossa solução é muito longa, e o limite é 250 bytes. Então, precisamos reescrever o programa nesse limite!