Exercícios: Pilhas

Ronaldo Fumio Hashimoto

1. Sejam os inteiros 1, 2, 3 e 4 que são lidos nesta ordem para serem colocados numa pilha. Considerando-se todas as possíveis sequências de operações *Empilha* e *Desempilha* decida quais das 24 (4!) permutações possíveis podem ser obtidas como saída da pilha. Por exemplo, a permutação 2, 3, 1, 4 pode ser obtida da seguinte forma: *Empilha* 1, *Empilha* 2, *Desempilha* 2, *Empilha* 3, *Desempilha* 3, *Desempilha* 1, *Empilha* 4, *Desempilha* 4.

2. Na sequência abaixo considere que uma letra significa *Empilha* e um '*' significa *Desempilha*:

   UTA*EC**R**O**

   Qual é a sequência em que as letras são desempilhadas?

3. Usando a notação do exercício anterior, é possível incluir '*' nas sequências abaixo para produzir a palavra ALGORITMO como resultado?

   - LGAROMOTI
   - ALGORITMO
   - OMTIROGLA
   - OTAIGLROM

4. Seja P o seguinte conjunto de cadeia sobre $\{a, b, c\}$:

   $$P = \{c, aca, bca, abcba, bacab, bbcbb, \ldots\}$$

   Uma cadeia deste conjunto pode ser especificada por $\alpha c \alpha^{-1}$, onde α é uma sequência de letras que são contém a's e b's e α^{-1} é o reverso de α, ou seja, α lido de trás para frente.

   Dada uma cadeia β, faça um programa que determina se β pertence ou não a P, ou seja, determina se β é da forma $\alpha c \alpha^{-1}$ para alguma cadeia α.

5. Escreva um algoritmo, usando uma *Pilha*, que inverte as letras de cada palavra de um texto preservando a ordem das palavras. Por exemplo, dado o texto:

   ESTE EXERCÍCIO É MUITO FÁCIL

   a saída deve ser

   ETSE OICÍCREXE É OTIUM LICÁF

6. Simule a execução do algoritmo de conversão para a notação posfixa com a expressão aritmética abaixo:

   $$(A + B) * D + E/(F + A * D) + C$$

7. Considerando o algoritmo de conversão de notação infixa para posfixa responda às seguintes perguntas.

   - Qual é o tamanho máximo que a pilha pode atingir se a expressão a ser traduzida tiver tamanho n (i.e., o numero total de operandos, operadores, e abre e fecha parêntesis na expressão é n)? Você pode supor que a expressão está correta.
   - Qual é a resposta para o item anterior se restringirmos o número de parêntesis na expressão para no máximo 6 (número de pares abre e fecha parêntesis)?

8. Uma outra forma de expressão sem parêntesis que é fácil de ser avaliada é chamada de notação prefixa. Nesta forma de escrever as expressões aritméticas os operadores precedem seus operandos. Por exemplo:

Infixa	Prefixa
$A * B/C$	$/ * ABC$
$A * (D + C)/B - G$	$-/ * +DCBG$
$A + B * C - D * E + A * B$	$+ - +A * BC * DE * AB$

Observe que a ordem dos operandos não é alterada passando da notação infixa para a prefixa.

- Passe a expressão aritmética do exercício 6 para a notação prefixa.
- Escreva um algoritmo que transforma uma expressão na forma infixa para a expressão prefixa correspondente.

Sugestão: percorra a expressão infixa de trás para a frente.

9. Converta as seguintes expressões da notação infixa para a notação posfixa e prefixa, considerando também a operação de potenciação

   - `A^(B^(C*(D/(E-F))))+G`
   - `(A+B)*((C+D)*(E+F))^((G+H)*(I+J))*(K + L)`
   - `(A^B)^(C^D)`

10. Faça um programa que receba um string (sem caracteres acentuados) via linha de comando e imprima uma nova sequência de caracteres segundo as seguintes regras:

    - Toda não consoante é exibida diretamente;
    - Toda seqüência de consoantes é exibida na ordem inversa da sequência de entrada.

    Exemplo:

    ESTE EXERCICIO EH MUITO FACIL

    a saída deve ser

    ELCE EFETMIHIO EC CUIRO XATIS

11. **Empilhamento decrescente.** Uma empilhadeira carrega caixas de 7, 5, 3 toneladas. Há três pilhas A, B e C. A pilha A é onde se encontram todas as caixas que chegam no depósito. Com um detalhe: caixas maiores não podem ser empilhadas sobre caixas menores. Elabore uma função chamada

    ```
    void ChegaNoDeposito (int caixa_nova, Pilha *pilhaA, Pilha *pilhaB, Pilha *pilhaC);
    ```

    que efetue o controle das caixas, de forma que caso chegue uma caixa de maior peso do que uma que já está em A deva ser empilhada, então, todas as caixas que estão em A são movidas para as pilhas auxiliares B (contendo somente caixa de 5 toneladas) e C (contendo somente caixas de 3 toneladas) **até** que se possa empilhar a nova caixa. Depois, todas as caixas são movidas de volta para a pilha A.

    Considere a sua pilha como a estrutura:

    ```
    typedef struct _pilha {
      int * p;
      int t;
      int size;
    } Pilha;
    ```

    Escreva e utilize as funções com os seguintes protótipos:

    ```
    int  vazia (Pilha *pilha);
    int  cheia (Pilha *pilha);
    void empilhar (int caixa, Pilha *Pilha);
    int  desempilhar (Pilha *pilha);
    int  topo (Pilha pilha);
    void destroi (Pilha *pilha);
    ```

12. Faça um programa que resolva o exercício que está em

    http://br.spoj.com/problems/EXPRES11/

    É possível submeter sua solução no *link* indicado acima.

    Pedrinho e Zezinho estão precisando estudar resolução de expressões matemáticas para uma prova que irão fazer. Para isso, eles querem resolver muitos exercícios antes da prova. Como sabem programar, então decidiram fazer um gerador de expressões matemáticas.

    O gerador de expressões que eles criaram funciona em duas fases. Na primeira fase é gerada uma cadeia de caracteres que contém apenas os caracteres '{', '[', '(', '}', ']' e ')'. Na segunda fase, o gerador adiciona os números e operadores na estrutura criada na primeira fase. Uma cadeia de caracteres é dita bem definida (ou válida) se atende as seguintes propriedades:

- Ela é uma cadeia de caracteres vazia (não contém nenhum caractere).
- Ela é formada por uma cadeia bem definida envolvida por parênteses, colchetes ou chaves. Portanto, se a cadeia S é bem definida, então as cadeias (S), $[S]$ e $\{S\}$ também são bem definidas.
- Ela é formada pela concatenação de duas cadeias bem definidas. Logo, se as cadeias X e Y são bem definidas, a cadeia XY é bem definida.

Depois que Pedrinho e Zezinho geraram algumas expressões matemáticas, eles perceberam que havia algum erro na primeira fase do gerador. Algumas cadeias não eram bem definidas. Eles querem começar a resolver as expressões o mais rápido possível, e sabendo que você é um ótimo programador, resolveram pedir que escreva um programa que dadas várias cadeias geradas na primeira fase, determine quais delas são bem definidas e quais não são.

Entrada

A entrada é composta por diversas instâncias. A primeira linha da entrada contém um inteiro T indicando o número de instâncias. Em seguida temos T linhas, cada uma com uma cadeia A.

Saída

Para cada instância imprima uma linha contendo a letra S se a cadeia é bem definida, ou a letra N, caso contrário.

Restrições

- $1 \leq T \leq 20$.
- a cadeia de caracteres A tem entre 1 e 100000 caracteres.
- a cadeia de caracteres A contém apenas caracteres '{', '[', '(', '}', ']' e ')'.

Exemplo de entrada

```
12
()
[]
{}
(]
}{
([{}])
{}()[]
()]
{[]
(
(([{}{}()[]])(){})
((((((((((){([])}]))))))))))
```

Exemplo de saída

```
S
S
S
N
N
S
S
N
N
N
S
N
```

13. Faça um programa que resolva o exercício que está em

    http://www.spoj.com/problems/STPAR/

    É possível submeter sua solução no *link* indicado acima.

Segue o enunciado em “Inglês”

For sure, the love mobiles will roll again on this summer's street parade. Each year, the organisers decide on a fixed order for the decorated trucks. Experience taught them to keep free a side street to be able to bring the trucks into order.

The side street is so narrow that no two cars can pass each other. Thus, the love mobile that enters the side street last must necessarily leave the side street first. Because the trucks and the ravers move up closely, a truck cannot drive back and re-enter the side street or the approach street.

You are given the order in which the love mobiles arrive. Write a program that decides if the love mobiles can be brought into the order that the organisers want them to be.

Input

There are several test cases. The first line of each test case contains a single number n, the number of love mobiles. The second line contains the numbers 1 to n in an arbitrary order. All the numbers are separated by single spaces. These numbers indicate the order in which the trucks arrive in the approach street. No more than 1000 love mobiles participate in the street parade. Input ends with number 0.

Output

For each test case your program has to output a line containing a single word yes if the love mobiles can be re-ordered with the help of the side street, and a single word no in the opposite case.

Sample input

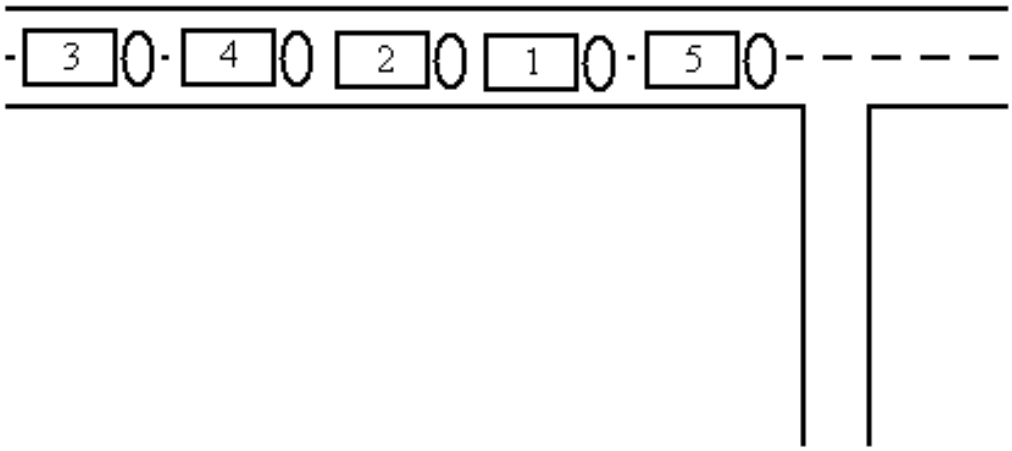
```
5
5 1 2 4 3
0
```

Sample output

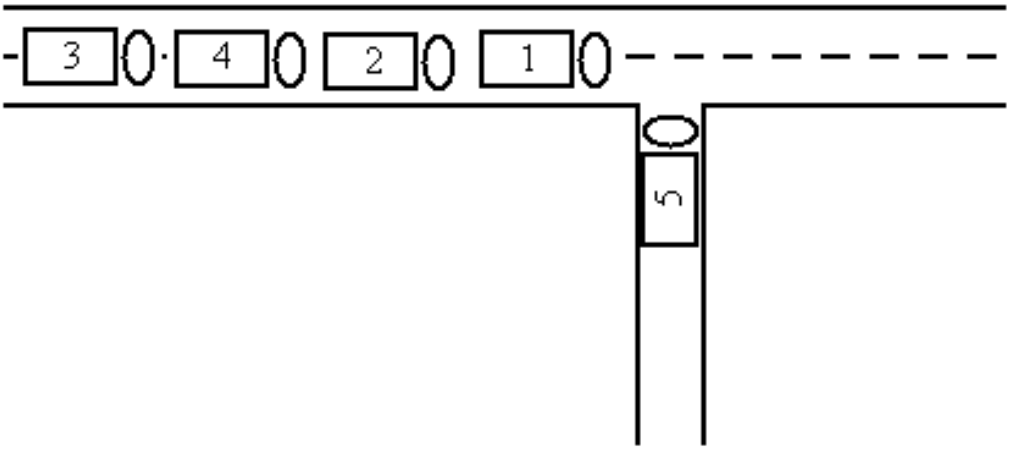
```
yes
```

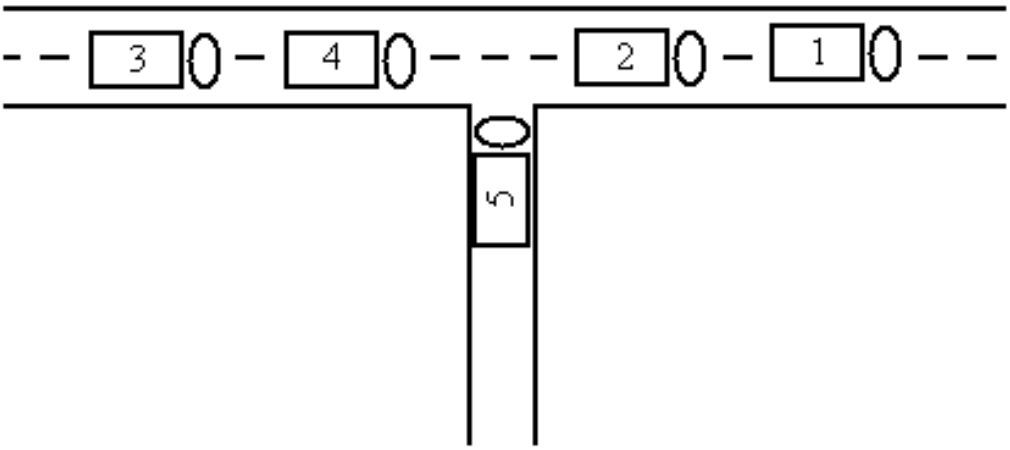
Illustration

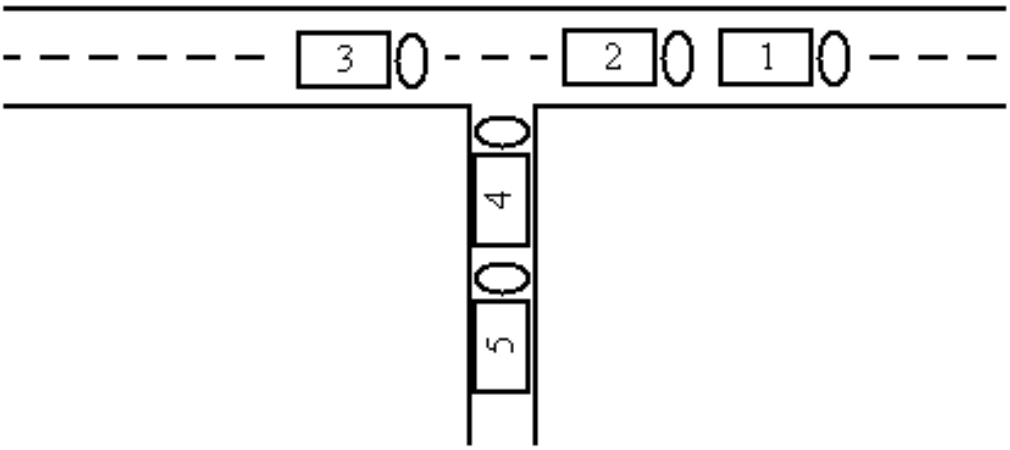
The sample input reflects the following situation:

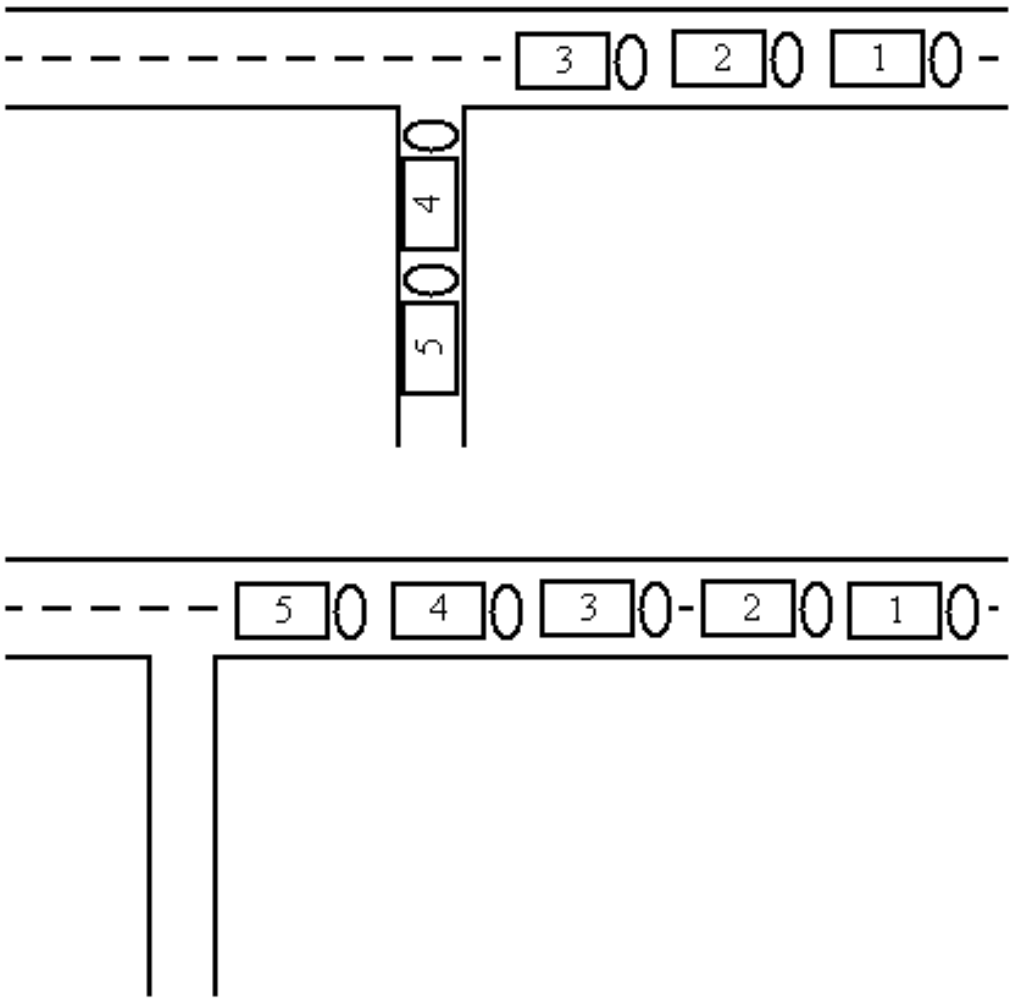


The five trucks can be re-ordered in the following way:


3
4
2
1
5


3
4
2
1
5


3
2
1
4
5



14. Faça um programa que resolva o exercício que está em

https://www.codechef.com/problems/COMPILER

Segue o enunciado em “Inglês”.

Lira is now very keen on compiler development. :)

She knows that one of the most important components of a compiler, is its parser. A parser is, in simple terms, a software component that processes text, and checks it’s semantic correctness, or, if you prefer, if the text is properly built.

As an example, in declaring and initializing an integer, in C/C++, you can’t do something like:

```
int = x ;4
```

as the semantics of such statement is incorrect, as we all know that the datatype must precede an identifier and only afterwards should come the equal sign and the initialization value, so, the corrected statement should be:

```
int x = 4;
```

Today, Lira is concerned with an abstract instruction which is composed of the characters “<” and “>” , which she will use on the design of her language, L++ :D.

She is using it as an abstraction for generating XML code Tags in an easier fashion and she understood that, for an expression to be valid, a "<" symbol must always have a corresponding ">" character somewhere (not necessary immediately) after it. Moreover, each ">" symbol should correspond to exactly one "<" symbol. So, for instance, the instructions:

```
<<>>
<>
<><>
```

are all valid. While:

```
>>
><><
```

are not.

Given some expressions which represent some instructions to be analyzed by Lira's compiler, you should tell the length of the longest prefix of each of these expressions that is valid, or 0 if there's no such a prefix.

Input

Input will consist of an integer T denoting the number of test cases to follow. Then, T strings follow, each on a single line, representing a possible expression in L++.

Output

For each expression you should output the length of the longest prefix that is valid or 0 if there's no such a prefix.

Constraints

- $1 \leq T \leq 500$;
- $1 \leq$ The length of a single expression ≤ 106;
- The total size all the input expressions is no more than $5 * 106$.

Sample input

```
3
<<>>
><
<>>>
```

Sample output

```
4
0
2
```