**Problema A: Amanhã Eu Faço**

*TL: 2 segundos*

*ML: 128 MB*

Como todo estudante, Jonas gosta de procrastinar. Ele recebeu uma lista de números inteiros, e deve imprimí-los na tela. Porém, ao invés de imprimir todos de uma só vez, ele os imprime aos poucos.

Resumindo, Jonas decide que a cada **K** números ele vai imprimir os **D** primeiros, e o resto ele vai descartar (na verdade ele vai imprimí-los amanhã). Sua tarefa é simular o que Jonas fez.

Por exemplo, considere o caso em que **K** = 3 e **D** = 2. Se na entrada tivermos a seguinte sequência de números: **1**, **2**, 5, **1**, **4**, 0, **3**; o seu programa deve imprimir: 1, 2, 1, 4, 3. Mesmo que a sequência não termine, seu programa deve imprimir os números.

**Entrada**

A primeira linha da entrada contém três números, **N** (1 <= **N** <= 10000), **K** (1 <= **K** <= **N**) e **D** (1 <= **D** <= **K**). **N** números seguirão na próxima linha, separados por espaços em branco. Estes números vão de 0 até 2^31-1. Você deve ler até o **EOF.**

**Saída**

Para cada caso de teste imprima em linhas distintas os números imprimidos por Jonas.

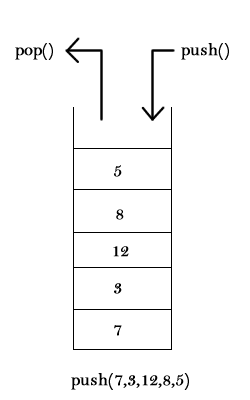
|  |  |
| --- | --- |
| **Entrada de Teste**  7 3 2  1 2 5 1 4 0 3  3 2 1  1 2 3  2 1 1  1 1 | **Saída de Teste**  1  2  1  4  3  1  3  1  1 |

**Problema B: Bão Dimais**

*TL: 2 segundos*

*ML: 128 MB*

O restaurante Bão Dimais precisa que você projete um algoritmo capaz de solucionar o problema de empilhamento de pratos sujos. Você receberá uma lista de números inteiros representando os pratos, na ordem em que eles foram colocados em cima da mesa. Seu objetivo é imprimir a ordem em que os pratos seriam retirados para serem lavados.



**Entrada**

A primeira linha da entrada contém um número **N** (1 <= **N** <= 10000), a quantidade de pratos. **N** números seguirão na próxima linha, separados por espaços em branco. Estes números vão de 0 até 2^31-1. Você deve ler até o **EOF.**

**Saída**

Imprima para cada caso de teste a ordem pedida, imprima um número em cada linha.

|  |  |
| --- | --- |
| **Entrada de Teste**  3  1 2 3  5  7 3 12 8 5 | **Saída de Teste**  3  2  1  5  8  12  3  7 |

**Problema C: Caixa Do Banco**

*TL: 2 segundos*

*ML: 128 MB*

Em um determinado banco, as pessoas recebem uma senha que determina a ordem em que elas são atendidas. O trabalho até agora era manual, e o gerente está te pagando para automatizar o processo.

O sistema deve funcionar assim. Você receberá dois tipos de comando:

* **CHEGOU x** : este comando significa que a pessoa identificada pelo número **x** (1 <= **x** <= 2^31-1) chegou. Pode ter pessoas com mesmo número de identificação na fila! Para cada comando deste você deve imprimir o tamanho da fila depois que esta pessoa chegou.
* **ATENDER** : este comando significa que você deve atender a primeira pessoa da fila. Para cada comando deste você deve imprimir o número da pessoa atendida, e se a fila estiver vazia imprima 0.

Implemente o sistema pedido.

**Entrada**

Cada caso de teste começa com um número **N** (1 <= **N** <= 10000), que indica o número de comandos. Cada comando será do tipo indicado na descrição do problema. Você deve ler até o **EOF.**

**Saída**

Imprima os números de acordo com os comandos pedidos.

|  |  |
| --- | --- |
| **Entrada de Teste**  4  CHEGOU 1  CHEGOU 1  ATENDER  ATENDER  3  CHEGOU 1000  ATENDER  ATENDER | **Saída de Teste**  1  2  1  1  1  1000  0 |

**Problema D: Diego Na Fila**

*TL: 2 segundos*

*ML: 128 MB*

Mais um problema de filas, que chatisse.. Pessoas estão aguardando em uma fila. Porém as pessoas mais velhas são atendidas primeiro. Diego quer saber se ele será o último a ser atendido, você pode responder isto para ele?

Como Diego chegou por último, se alguma pessoa tiver a mesma idade, Diego será atendido depois.

**Entrada**

A entrada contém vários casos de teste. Cada caso começa com dois números inteiros **N** (1 <= **N** <= 10000) e **I** (1 <= **I** <= 100), que identifica quantas pessoas temos na fila e a idade de Diego. A seguir, **N** números inteiros seguirão separados por espaços. Você deve ler até o **EOF.**

**Saída**

Para cada número da entrada imprima **S** se Diego será o último ou **N** caso contrário.

|  |  |
| --- | --- |
| **Entrada de Teste**  3 1  1 2 3  3 2  1 2 3 | **Saída de Teste**  S  N |

**Problema E: Entre Strings**

*TL: 2 segundos*

*ML: 128 MB*

Você até agora achou que os problemas estão ordenados por dificuldade, não? Só que não. Este é o mais fácil da prova. Dado duas cadeias de caracteres imprima a concatenação das duas!!

**Entrada**

A entrada contém vários casos de teste. Cada caso tem duas strings separadas por espaço, contendo apenas letras do alfabeto. As strings terão tamanho máximo de 10000. Você deve ler até o **EOF.**

**Saída**

Para cada número da entrada imprima a concatenação das duas strings.

|  |  |
| --- | --- |
| **Entrada de Teste**  sim nao  tchau pardal | **Saída de Teste**  simnao  tchaupardal |

**Problema F: Fácil Ou Difícil?**

*TL: 2 segundos*

*ML: 128 MB*

É dado uma string (cadeia de caracteres) consistindo apenas de parenteses (). Uma string é balanceada se ocorre o correto casamento dos parenteses. Por exemplo, “(((())))” é balanceada enquanto que “))())” não é.

Escreva um programa que leia uma string deste tipo e decida se a string é ou não balanceada.

**Entrada**

Cada linha conterá uma string de apenas parênteses de tamanho máximo 10000 e no mínimo 1. Você deve ler até o **EOF.**

**Saída**

Para cada caso, imprima **S** se for balanceada ou **N** caso contrário.

|  |  |
| --- | --- |
| **Entrada de Teste**  (  )(  ()  ()((()))  (((())))  ))())  ()()()()()  (((()()()())))  (((((((((((((  (((((((())))) | **Saída de Teste**  N  N  S  S  S  N  S  S  N  N |

**Problema G: Gostei Desse Problema**

*TL: 2 segundos*

*ML: 128 MB*

É dado uma lista de palavras no dicionário. Dado algumas strings, imprima se elas existem ou não no dicionário.

**Entrada**

O problema tem um único caso de teste. Na primeira linha temos um número **N** (1 <= **N** <= 100000), que indica quantas palavras temos no dicionário. Nas **N** linhas seguintes teremos as palavras listadas uma a uma, de tamanho máximo 100. Depois teremos um número **Q** (1 <= **Q** <= 100000), que indica quantas strings devem ser procuradas no dicionário. Nas **Q** linhas seguintes teremos as palavras a serem procuradas, de tamanho máximo 100.

Todas as palavras contém apenas letras do alfabeto.

**Saída**

Para cada palavra a ser procurada imprima **S** se ela existe ou **N** caso contrário.

|  |  |
| --- | --- |
| **Entrada de Teste**  11  usem  hahaha  map  de  strings  asa  senao  BoLa  vai  dar  TLE  6  asa  bola  BoLa  DE  dE  vaia | **Saída de Teste**  S  N  S  N  N  N |