**Problema A: Amores de Carlota**

*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Carlota é uma menina que ama muitos meninos. Mas ela gosta mais de uns do que de outros. Por isso, fez uma lista com os nomes dos meninos de sua turma e atribuiu uma nota de 1 a 10 para cada um deles (e nunca repete uma nota).

Um belo dia, Filipe encontrou esta lista. Como Filipe gosta de fazer bullying com Carlota, ele disponibilizou a lista na internet, sendo que cada pessoa pode consultar seu nome e saber em qual posição da lista da Carlota ela está.

Filipe precisa de sua ajuda para implementar uma busca eficiente para o seu site.

**Entrada**

A primeira linha de cada caso de teste contém o número **N** (1 <= **N** <= 10^3) de meninos que Carlota deu nota. Para cada **N** linhas a seguir, existe um nome (sem espaços, de tamanho máximo 20) e uma nota atribuída a ele, esta nota é única para cada menino.

Após isso, haverá um número inteiro **M** (1 <= **M** <= 10^4), representando quantas consultas de nome serão realizadas. As próximas M linhas contêm nomes a serem consultados. Você deve ler até o **EOF.**

**Saída**

Para cada consulta, imprima a posição que a pessoa consultada se encontra na lista. Caso ela não esteja na lista, imprima “Carlota nao te ama!”.

|  |  |
| --- | --- |
| **Entrada de Teste**  2  Bruno 10.00  Anderson 8.75  5  Dieguinho  Sanchosa  Bruno  Joana  Anderson  1  oi 0.0  1  OI | **Saída de Teste**  Carlota nao te ama!  Carlota nao te ama!  1  Carlota nao te ama!  2  Carlota nao te ama! |

**Problema B: Matrix**

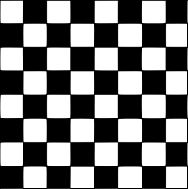
*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Você já teve um sonho que você tinha quase certeza que era real? E se você não fosse capaz de acordar deste sonho? Como você saberia a diferença entre o mundo do sonho e o mundo real?

Para desvendar esta complexa pergunta, uma das perguntas que devem ser respondidas é se o mundo que você vive pode ser representado por uma matriz em forma de xadrez.

Uma matriz de **N** linhas e **M** colunas está na forma de xadrez se e somente se os 4 vizinhos de uma célula são de uma cor diferente. Os vizinhos são as células na esquerda, direita, acima e abaixo da célula atual (com exceção das bordas é claro). As únicas cores aqui representadas serão 1 para branco e 0 para preto.



**Entrada**

A primeira linha de cada caso de teste contém os números **N** (1 <= **N** <= 10^3) e **M** (1 <= **M** <= 10^3), indicando o número de linhas e colunas respectivamente. Para cada **N** linhas a seguir, haverá **M** números (1 para branco e 0 para preto).

Você deve ler até o **EOF.**

**Saída**

Para cada caso imprima **S** se a matriz está em forma de xadrez ou **N** caso contrário.

|  |  |
| --- | --- |
| **Entrada de Teste**  2 2  1 0  0 1  1 1  0  1 1  1  3 3  1 0 1  1 0 1  0 1 0  3 3  1 0 1  0 1 0  1 0 1  3 3  0 1 0  1 0 1  0 1 0  3 3  0 1 0  1 0 1  0 1 1 | **Saída de Teste**  S  S  S  N  S  S  N |

**Problema C: Fibonacci**

*TL: 1 segundo (~1000 casos)*

*ML: 64 MB*

A sequência de Fibonacci é muito famosa na matemática. A sequência recebeu o nome do matemático italiano Leonardo de Pisa, mais conhecido por Fibonacci (contração do italiano filius Bonacci), que descreveu, no ano de 1202, o crescimento de uma população de coelhos, a partir desta.

Os números de Fibonacci são, portanto, os números que compõem a seguinte sequência: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, … Em termos matemáticos, a sequência é definida recursivamente pela fórmula abaixo, sendo o primeiro termo *F*1= 1:

O objetivo deste problema é calcular o i-ésimo número da sequência modulo 10^9+7, ou seja, em C: (Fi = Fi % (10^9+7)).

**Entrada**

Há vários casos de teste. Cada caso contém um único número **N** (1 <= **N** <= 10^6). Você deve ler até o **EOF.** Atenção!! Este problema tem por volta de 1000 casos de teste!!

**Saída**

Para cada caso imprima o i-ésimo número de Fibonacci.

|  |  |
| --- | --- |
| **Entrada de Teste**  1  2  3  13  100  1000000 | **Saída de Teste**  1  1  2  233  687995182  918091266 |

**Problema D: Exponenciando**

*TL: 1 segundo (~1000 casos)*

*ML: 64 MB*

Em várias contas matemágicas, precisamos calcular um número exponenciado à outro, ou seja, calcular **A^B = C**. Para tal tarefa temos em C a função **pow** da lib **math.h**. Porém, em certas contas númericas precisamos do cálculo deste valor módulo algum outro número **D**.

Sua tarefa aqui é calcular a seguinte equação, dados **A, B** e **D**:

**A^B = C mod D**

Lembre de aula de Quarta, onde foram mostrados três códigos de como exponenciar um número. Se você não lembra ou não foi à aula (por que não? :( ) chame um monitor.

**Entrada**

Há vários casos na entrada. Cada caso você deve ler três inteiros, **A, B** (1 <= **A,B** <= 10^9) e **D** (1 <= **D** <= 10^4). Você deve ler até o **EOF.**

**Saída**

Imprima em uma única linha o valor de **C**.

|  |  |
| --- | --- |
| **Entrada de Teste**  2 3 10  2 3 1  3 3 20  2 20 2  1000000000 1000000000 10000 | **Saída de Teste**  8  0  7  0  0 |

**Problema E: Enchendo a Mochila**

*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Você tem em mãos uma mochila com certa capacidade **C**, um peso máximo que ela aguenta de objetos. Em cima de uma mesa há vários objetos cada um com um peso. Seu objetivo é calcular o valor do maior peso de um sub-conjunto que não exceda a capacidade da mochila.



**Entrada**

A primeira linha de cada caso de teste contém o número **N** (1 <= **N** <= 20), o número de objetos, e o número **C** (1 <= **C** <= 10^9). A linha seguinte possui **N** valores. Cada valor é o peso dos objetos (0 <= **valor** <= 10^7). Você deve ler até o **EOF.**

**Saída**

Imprima em uma única linha o valor do maior peso de um sub-conjunto que não exceda a capacidade da mochila.

|  |  |
| --- | --- |
| **Entrada de Teste**  3 2  1 1 1  4 11  1 2 3 4  3 10  5 4 8 | **Saída de Teste**  2  10  9 |

**Problema F: Números Bonitos**

*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Um número bonito consiste de apenas dígitos 4 e 7. Por exemplo, o número 447 é bonito. Um número muito bonito é um número bonito onde a quantidade de dígitos 4 é igual a quantidade de dígitos 7. Por exemplo, os números 47 e 4747 são muito bonitos. Você possui um número em mãos. Qual o menor número muito bonito que é maior ou igual que o número que você tem em mãos?

**Entrada**

A primeira linha de cada caso de teste contém o número **N** (1 <= **N** <= 10^9), o número que Fossa possui em mãos. Você deve ler até o **EOF.**

**Saída**

Imprima em uma única linha o menor número bonito maior ou igual a **N**. Note que a resposta pode ser maior que o valor máximo de **N**.

|  |  |
| --- | --- |
| **Entrada de Teste**  4500  47  1  99 | **Saída de Teste**  4747  47  47  4477 |

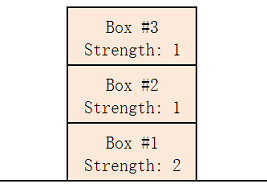
Robinho, um cachorro muito sabido, disse que é necessário usar ‘**long long**’ ao invés de ‘**int**’, porque a resposta pode não caber em um inteiro. Para imprimir ou ler um ‘**long long**’ use o cout e cin! Se você estiver usando C no linux pode usar printf(“%Ld\n”, numero) ou scanf(“%Ld”, &numero). Robinho é um anjo que já mora no céu.

**Problema G: Fossa o Macaco**

*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Fossa é um macaco de circo muito alegre que empilha caixas. Fossa possui um número de caixas a empilhar verticalmente, uma em cima da outra. Cada caixa aguenta até um certo número máximo de caixas empilhadas em cima dela, depois deste valor a caixa quebra e a pilha entra em colapso. Ele precisa formar o menor número possível de pilhas respeitando as limitações das caixas, ou seja, sem elas entrarem em colapso. Você pode ajudá-lo? (Não vale responder não)..



Para ficar mais claro olhe os exemplos de casos de teste dados na próxima página.

**Entrada**

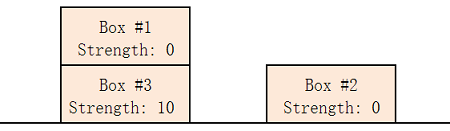
A primeira linha de cada caso de teste contém o número **N** (1 <= **N** <= 100), o número de caixas. A linha seguinte possui **N** valores. Cada valor, é a quantidade de caixas que uma caixa aguenta (0 <= valor <= 100). Você deve ler até o **EOF.**

**Saída**

Imprima em uma única linha o menor número de pilhas que Fossa consegue formar.

|  |  |
| --- | --- |
| **Entrada de Teste**  3  0 0 10  5  0 1 2 3 4  4  0 0 0 0  9  0 1 0 2 0 1 1 2 10 | **Saída de Teste**  2  1  4  3 |

Para o primeiro caso, Fossa o macaco teria que formar 2 pilhas:



Para o segundo caso, Fossa o macaco teria que formar 1 pilha:

