

## Problema A

### Busca

*Nome base: busca*

*Tempo limite: 1s*

A Nlogônia ainda quer criar um cassino. Continua querendo ser uma LasVegas(n)!!! É importante preparar a cidade para receber os clientes do cassino. Ela quer mapear e ajudar as pessoas a encontrarem determinados locais. Cada local é indicado por um número, por exemplo, Rua do Cassino é o número 5, Rua da farmácia é o número 8 e assim toda a cidade é identificada por números. Porém, ainda não mapeou todos os locais da cidade. Você foi convidado a mostrar para as pessoas se o local que ela procura está mapeado ou não.

#### ENTRADA

A primeira linha contém números inteiros indicando os locais mapeados. A segunda linha contém os números que serão verificados pelos clientes.

#### SAÍDA

Para cada caso de teste imprima uma mensagem se está mapeado ou não.

Exemplo de Entrada	Exemplo de Saída
2 4 6 8 10 1 4	1 Não está mapeado 4 Está mapeado
3 6 8 0 123 123 45 67	123 Está mapeado 45 Não está mapeado 67 Não está mapeado

## Problema B

### Ordenação de Pontos por Distância

*Nome base: distancia*

*Tempo limite: 1s*

Solicitaram para que você desenvolvesse um programa que ordena um conjunto de pontos no plano cartesiano, com base em sua distância à origem (ponto de referência) em ordem crescente. A distância entre dois pontos  $(x_1, y_1)$  e  $(x_2, y_2)$  é calculada com a raiz quadrada da soma dos quadrados das diferenças nas coordenadas, ou seja: distância =  $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$ . Para evitar cálculos de raiz quadrada (que é uma operação custosa), você pode comparar as distâncias ao quadrado, já que isso preserva a ordem relativa dos pontos. Portanto, você pode calcular e comparar as distâncias ao quadrado para determinar a ordem de classificação dos pontos.

#### ENTRADA

A primeira linha deve conter um inteiro  $N$  ( $1 \leq N \leq 100$ ), que representa o número de pontos. As próximas  $N$  linhas contêm as coordenadas  $x$  e  $y$  de cada ponto, separadas por um espaço.

#### SAÍDA

A saída deve conter as coordenadas dos pontos ordenados com base em sua distância à origem, uma coordenada por linha.

Exemplo de Entrada	Exemplo de Saída
5 1 1 2 2 3 3 0 0 -1 -1	0 0 1 1 -1 -1 2 2 3 3

Exemplo de Entrada	Exemplo de Saída
6 3 4 1 2 -2 2 0 0 5 1 -3 -5	0 0 1 2 -2 2 3 4 5 1 -3 -5



**Nota:** em caso de empate (ou seja pontos valores diferentes e a mesma distância), use os pontos na ordem em que foram recebidos.

## Problema C

### Ordenação de Números Impares e Pares

*Nome base: numeros*

*Tempo limite: 1s*

Marcos estava estudando sobre algoritmos de ordenação e testou algumas implementações eficientes para ordenar números inteiros. Após realizar algumas implementações Marcos ainda não estava satisfeito com o seu aprendizado, foi nesse momento que ele teve a ideia de construir um algoritmo de ordenação que receba uma lista de números inteiros e os ordena de tal forma que os números ímpares apareçam primeiro na lista ordenados entre si, seguindo pelos números pares também ordenados entre si, mantendo assim a ordem relativa entre as sequências. Ajude o Marcos com a implementação do programa.

#### ENTRADA

A primeira linha deve conter um inteiro  $N$  ( $1 \leq N \leq 100$ ), que representa a quantidade de números na lista. A segunda linha contém um  $N$  números inteiros separados por espaços, representando a lista de números.

#### SAÍDA

A saída deve conter os números ordenados de acordo com as regras acima, separados por espaços.

Exemplo de Entrada	Exemplo de Saída
8 3 7 1 4 8 6 2 5	1 3 5 7 2 4 6 8

Exemplo de Entrada	Exemplo de Saída
10 3 7 1 4 8 6 2 5 9 10	1 3 5 7 9 2 4 6 8 10

## Problema D

### O Brilho Começa

Nome base: *brilho*

Tempo limite: 1s

Imagine que você tem  $n$  lâmpadas numeradas de  $1, 2, \dots, n$ . Inicialmente, todas as lâmpadas estão ligadas. Alternar o estado de uma lâmpada significa desligá-la se estava ligada, ou ligá-la se estava desligada.

A seguir, você realiza o seguinte procedimento:

- Para cada  $i=1, 2, \dots, n$ , alterne o estado de todas as lâmpadas  $j$  tal que  $j$  seja divisível por  $i$ .

Após realizar todas as operações, várias lâmpadas ainda estarão ligadas. Seu objetivo é fazer com que o número de lâmpadas ligadas seja exatamente  $k$ .

Encontre o menor valor de  $n$  tal que, após realizar todas as operações, haverá exatamente  $k$  lâmpadas ligadas. Podemos garantir que sempre existe uma resposta.

#### Entrada

Cada teste contém múltiplos casos de teste. A primeira linha contém o número de casos de teste  $t$  ( $1 \leq t \leq 10^4$ ). A descrição dos casos de teste segue.

A única linha de cada caso de teste contém um único inteiro  $k$  ( $1 \leq k \leq 10^{18}$ ).

#### Saída

Para cada caso de teste, exiba  $n$  — o menor número de lâmpadas.

Exemplo de Entrada	Exemplo de Saída
3 1 3 8	2 5 11

#### Explicação

No primeiro caso de teste, o menor número de lâmpadas é  $n = 2$ . Vamos representar o estado de todas as lâmpadas com um array, onde 1 representa uma lâmpada ligada e 0 representa uma lâmpada desligada. Inicialmente, o array é  $[1, 1]$ .

1. Após realizar a operação com  $i = 1$ , o array se torna  $[0, 0]$ .
2. Após realizar a operação com  $i = 2$ , o array se torna  $[0, 1]$ .

Ao final, há  $k = 1$  lâmpada ligada. Podemos mostrar que a resposta não pode ser menor que 2.

No segundo caso de teste, o menor número de lâmpadas é  $n = 5$ . Inicialmente, o array é  $[1, 1, 1, 1, 1]$ .

1. Após realizar a operação com  $i = 1$ , o array se torna  $[0, 0, 0, 0, 0]$ .
2. Após realizar a operação com  $i = 2$ , o array se torna  $[0, 1, 0, 1, 0]$ .
3. Após realizar a operação com  $i = 3$ , o array se torna  $[0, 1, 1, 1, 0]$ .
4. Após realizar a operação com  $i = 4$ , o array se torna  $[0, 1, 1, 0, 0]$ .
5. Após realizar a operação com  $i = 5$ , o array se torna  $[0, 1, 1, 0, 1]$ .

Ao final, há  $k = 3$  lâmpadas ligadas.