

## B. Tabela de Hash

time limit per test: 1 second  
 memory limit per test: 256 megabytes

Tabelas de Hash, também conhecidas como tabelas de dispersão, armazenam elementos com base no valor absoluto de suas chaves e em técnicas de tratamento de colisões. Para o cálculo do endereço onde deve ser armazenada uma determinada chave, utiliza-se uma função denominada função de dispersão, que transforma a chave em um dos endereços disponíveis na tabela.

Suponha que uma aplicação utiliza uma tabela de dispersão com 13 endereços-base (índices variam de 0 a 12) e emprega a função de dispersão  $h(x) = x \bmod 13$ , onde  $\bmod$  é a operação que calcula o resto da divisão de  $x$  por 13, e  $x$  representa a chave do elemento cujo endereço-base deve ser calculado.

Se  $x$  for igual a 49, a função de dispersão retornará o valor 10, indicando o local onde esta chave deverá ser armazenada. Se a mesma aplicação considerar a inserção da chave 88, o cálculo retornará o mesmo valor 10, ocorrendo neste caso uma colisão. O Tratamento de colisões serve para resolver os conflitos nos casos onde mais de uma chave é mapeada para um mesmo endereço-base da tabela. Este tratamento pode considerar, ou o recálculo do endereço da chave ou o encadeamento dos valores naquela posição (também chamado de encadeamento exterior).

Nesta questão, sua tarefa é escrever um programa que calcula o endereço para inserções de diversas chaves em algumas tabelas, com funções de dispersão e tratamento de colisão por encadeamento exterior.

### Input

A entrada contém um caso de teste único. A primeira linha da entrada contém um valor  $M$  ( $1 \leq M \leq 1000$ ) que indica a quantidade de endereços-base na tabela seguido por um espaço e um valor  $C$  ( $1 \leq C \leq 2000$ ) que indica a quantidade de chaves a serem armazenadas. A segunda linha contém cada uma das chaves  $V$  ( $0 \leq V \leq 10000$ ), separadas por um espaço em branco.

### Output

A saída deve conter  $M$  linhas, onde cada linha irá informar a posição da estrutura e as chaves que foram armazenadas para cada uma, terminando com o caracter ' ' e todos separados por "->", conforme os exemplos abaixo.

### Examples

input	Copy
13 9 44 45 49 70 27 73 92 97 95	
output	Copy
0 -> \ 1 -> 27 -> 92 -> \ 2 -> \ 3 -> \ 4 -> 95 -> \ 5 -> 44 -> 70 -> \ 6 -> 45 -> 97 -> \ 7 -> \ 8 -> 73 -> \ 9 -> \ 10 -> 49 -> \ 11 -> \ 12 -> \	

input	Copy
7 8 35 12 2 17 19 51 88 86	
output	Copy

### IDP - TAA - 2025/02

Private

Participant



### → About Group



Este grupo tem o objetivo de organizar as atividades de programação da disciplina de Técnicas de Programação e Análise de Algoritmos.

[Group website](#)

### → Group Contests

- TAA - LEE 05
- TAA - LEA 04
- TAA - LEE 04
- TAA - AS 01
- TAA - LEA 03
- TAA - LEE 03
- TAA - LEA 02
- TAA - LEE 02
- TAA - LEA 01
- TAA - LEE 01
- ET - Exercício de Testes

### TAA - LEE 05

Contest is running

4 days

Contestant



### → Submit?

Language: GNU G++17 7.3.0 ▼

0 -> 35 -> \  
1 -> \  
2 -> 2 -> 51 -> 86 -> \  
3 -> 17 -> \  
4 -> 88 -> \  
5 -> 12 -> 19 -> \  
6 -> \

Choose  
file:

Escolher arquivo

Nenhu...colhido

Submit

---

[Codeforces](#) (c) Copyright 2010-2025 Mike Mirzayanov  
The only programming contests Web 2.0 platform  
Server time: Oct/29/2025 21:10:18<sup>UTC-3</sup> (k2).  
Desktop version, switch to [mobile version](#).  
[Privacy Policy](#) | [Terms and Conditions](#)

Supported by



**ITMO**