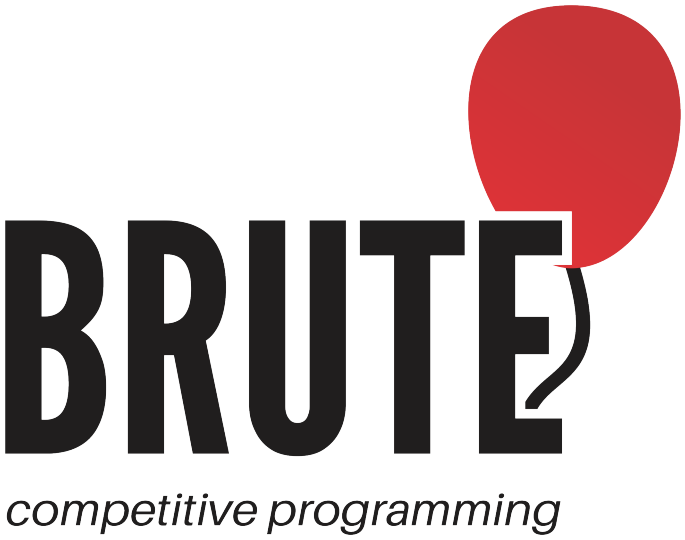
Caderno de Provas
Maratona CODECON
2020

Maratona de Programação - CODECON

Servidor BOCA:
http://boca.bruteudesc.com/

Organização e Realização:

BRUTE UDESC



Lembretes:

- Aos *javaneiros*: **o nome da classe deve ser o mesmo nome do arquivo a ser submetido**. Ex: **classe petrus**, nome do arquivo **petrus.java**;
- Exemplo de leitura de entradas que funcionam:

```
Java: (import java.util.Scanner)
Scanner in = new Scanner(System.in);
ou
Scanner stdin = new Scanner(new BufferedReader(new InputStreamReader(System.in)));

C: (#include <stdio.h>)
int integer1; scanf("%d", &integer1);

C++: (#include <iostream>)
int integer1; std::cin >> integer1;
```

  Exemplo de saída de entradas:

```
Java: System.out.format("%d %d\n", integer1, integer2);
C: printf("%d %d\n", integer1, integer2);
C++: std::cout << integer1 << " " << integer2 << std::endl;
```

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova;
- A correção é automatizada, portanto, **siga atentamente as exigências da tarefa quanto ao formato da entrada e saída conforme as amostras dos exemplos**. Deve-se considerar entradas e saídas padrão;
- Todos os compiladores (Java, Python, C e C++) são padrões da distribuição Ubuntu versão 16.04 (gcc C11 habilitado);
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas;
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos;
- Utilize o *clarification* para dúvidas da prova. Os juízes podem **opcionalmente** atendê-lo com respostas acessíveis a todos;
- Algumas interfaces estão disponíveis nas máquinas Linux, que podem ser utilizada no lugar da *Unity*. Para isto, basta dar *logout*, e selecionar a interface desejada. Usuário e senha: *udesc*;
- Ao pessoal local: **cuidado com os pés sobre as mesas para não desligarem nenhum estabilizador/computador de outras equipes**!

Agradecimentos

- Alan Turing, ateu e homossexual pai da ciência da computação;
- George Boole, 1100010 1110010 1100001 1100010 11000011 10101101 1110011 1110011 1101001 1101101 1101111;

Maratona de Programação CODECON - 2020

Maratona de Programação CODECON

CODECON 2020

Conteúdo

1 Problema A: Permutações

`Arquivo: A.[c|cpp|java|py]` `Tempo limite: 1 s`
`Autor: João Vitor Fröhlich`

Dada uma sequência finita de números S, dizemos que uma permutação é uma forma de representar essa sequência utilizando todos os números, sem repeti-los. Por exemplo, dado uma sequência de 5 números consecutivos $S = \{1, 2, 3, 4, 5\}$, algumas possíveis permutações são as sequências $\{1, 2, 3, 4, 5\}$, $\{1, 2, 4, 5, 3\}$, $\{5, 4, 3, 2, 1\}$, enquanto as sequências $\{1, 2, 3, 4\}$, $\{1, 2, 3, 4, 5, 6\}$, $\{1, 2, 3, 3, 5\}$ não são permutações de S.

Problema

Dado um número inteiro N, calcular o número de permutações possíveis em uma sequência de N números consecutivos.

Entrada

Será fornecido um único inteiro de entrada, N.

Restrições

- $1 \leq N \leq 10$.

Saída

Você deve imprimir um único inteiro, representando o número de possíveis permutações para uma sequência de N elementos.

Exemplo de Entrada	**Exemplo de Saída**
3	6

No exemplo fornecido, as 6 permutações possíveis para uma sequência de 3 números são: $\{1, 2, 3\}$, $\{1, 3, 2\}$, $\{2, 1, 3\}$, $\{2, 3, 1\}$, $\{3, 1, 2\}$ e $\{3, 2, 1\}$.

2 Problema B: Eu pago quanto eu quero

`Arquivo: B.[c|cpp|java|py]` `Tempo limite: 1 s`
`Autor: Igor Schiessl Froehner`

Você trabalha em uma empresa de tecnologia que pesquisa e cataloga preços de lojas online, vocês estão criando uma nova *feature* em que o cliente diz especificamente qual preço ele quer pagar pelo produto e o aplicativo diz se existe tal preço, e se sim, em que loja ele se encontra. Como sua empresa faz análises dos dados, você já tem todos os preços das lojas ordenados, e é estranho, mas já é sabido que não existem preços repetidos também. Mas fique atento, pois são muitos dados!

Problema

Dado a lista dos preços ordenados de um produto X em uma loja, existe tal preço na lista? E se sim qual a posição desse preço na lista?

Entrada

Na primeira linha é dado N e M, respectivamente quantos preços há na lista, e quantos clientes querem fazer consulta em questão. Na próxima linha são dados os N preços p em ordem crescente da lista. Após isso vêm as M linhas com as consultas c dos clientes.

Restrições

- $1 \leq N, M \leq 1000000$.
- $0 \leq p, c \leq 100000000$.

Saída

Para cada uma das M consultas, você deve informar se tal preço existe na lista feita por sua empresa, e se existe, em qual posição ele se encontra. Se não existir a saída é -1.

Exemplo de Entrada	**Exemplo de Saída**

`3 2`
`1 2 3`
`1`
`9`
	`1 encontrado em 1`
	`-1`

`5 3`
`2 5 12 72 110`
`2`
`5`
`109`
	`2 encontrado em 1`
	`5 encontrado em 2`
	`-1`

3 Problema C: Joãozinho e seus LEDs

`Arquivo: C.[c|cpp|java|py]` `Tempo limite: 1 s`
`Autor: João Vitor Fröhlich`

Após algum tempo de quarentena, Joãozinho estava entediado em casa, e por isso decidiu ocupar sua mente com coisas inúteis.

Não contente em apenas pensar, decidiu construir um painel de LED, constituído por N fileiras, cada uma contendo M LEDs. Esses LEDs possuiam apenas 2 cores, vermelho e azul. Joãozinho montou o painel pensando em ligar todos os LEDs inicialmente na cor azul e, ao apertar um botão, os LEDs trocariam de cor para vermelho. Contudo, um dos LEDs estava trocado, de forma que um LED estava vermelho no painel, enquanto os outros estavam azuis, da mesma forma que quando todos estavam vermelhos, este estava azul.

Ao invés de modificar especificamente esse LED, Joãozinho teve uma idéia diferente, modificou a função do botão. Agora, quando o botão era acionado, ao invés de todos os LEDs trocarem de cor, apenas os LEDs adjacentes a um LED de cor diferente mudaria sua cor. Dessa forma, se um LED azul é adjacente a um LED vermelho, o LED azul viraria vermelho, enquanto o vermelho se tornaria azul, e da mesma forma ocorreria caso o LED vermelho for adjacente a um LED azul, como mostra a imagem a seguir.

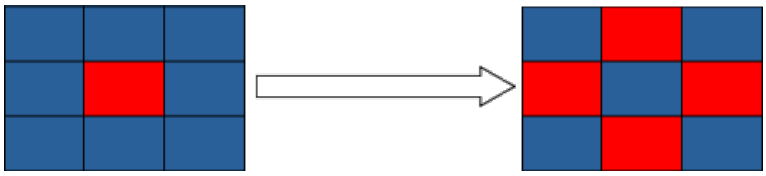


Figura 1: Comportamento do botão

Problema

Joãozinho então pensou: num painel de LEDs NxM, onde todos os LEDs começariam na cor azul e sabendo a posição do LED vermelho, como estaria o painel quando o botão fosse acionado p vezes?

Entrada

Na primeira linha, serão fornecidos três números inteiros N, M, p, respectivamente o número de fileiras, o número de LEDs em cada fileira e o número de vezes que p será apertado.

Na segunda linha, serão fornecidos dois inteiros x, y, correspondentes à posição do LED vermelho no painel. Nesse caso, x representa o número da fileira e y qual o número do LED na fileira.

Restrições

- $1 \leq N, M \leq 100$.
- $0 \leq p \leq 100$.
- $1 \leq x \leq n$.
- $1 \leq y \leq m$.

Saída

Deverá ser apresentado uma matriz, de tamanho $N \times M$, correspondente à distribuição dos LEDs ao final das operações. Para posições onde o LED for azul, deverá ser escrito o caractere 'A', enquanto para posições onde o LED for vermelho, deverá ser escrito o caractere 'V'.

Exemplo de Entrada	**Exemplo de Saída**
3 3 1 2 2	AVA VAV AVA
2 2 3 2 1	VA AV
4 4 1 4 1	AAAA AAAA VAAA AVAA

4 Problema D: Vales e Montanhas

`Arquivo: D.[c|cpp|java|py]` `Tempo limite: 1 s`
`Autor: Ferdinando Augusto Galera Junior`

Manuel é um geólogo que gosta de desenhar os relevos. Para facilitar sua vida ele contratou **VOCÊ** para criar um programa que desenha vales e montanhas com base em nas informações descritas por Manuel.
Primeiro é informado se Manuel quer desenhar um vale ou uma montanha, em seguida a altura do pico ou profundidade do vale, e por último a dimensão da matriz ($T \times T$).
A figura sempre deverá estar centralizada em relação ao pico/vale, preenchida por 0's com exceção da silhueta da montanha/vale que será representado por 1's.

Entrada

A entrada contém vários casos de teste e termina com EOF (*end of file*). Cada entrada consiste de 3 valores inteiros, Operação **O** (0 ou 1 onde representa vales ou montanha, respectivamente), Altura **A** e Tamanho da matriz **T**.

Restrições

- $0 \leq O \leq 1$.
- $1 \leq A \leq T$.
- $3 \leq T \leq 70$.

Saída

Para cada caso de teste, apresente a(s) figura(s) da montanha(s)/vale(s) separada(s) por quebra de linha.

Exemplo de Entrada	**Exemplo de Saída**
0 5 5	0 0 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 3 5	1 1 1 1 1 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 6 7 0 4 6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1

5 Problema E: N-nacci

`Arquivo: E.[c|cpp|java|py]` `Tempo limite: 2 s`
`Autor: Rafael Granza de Mello`

A sequência de **Tribonacci** começa com [1, 1, 1] e a partir daí todos os próximos termos são as somas dos **três** termos anteriores. A sequência até o sétimo termo ficaria assim [1, 1, 1, 3, 5, 9, 17].

Problema

Muitos acham essa sequência um tanto sem graça, e para dar uma variada vamos fazer um N-nacci! Isso quer dizer que um termo será a soma dos N termos anteriores e nos vamos querer saber qual será o I-ésimo termo dessa sequência. Note que é não é possível definir os primeiros N termos dessa forma, por isso nós vamos te dizer quais são eles.

Entrada

A entrada consiste de um inteiro N indicando quantos termos deverão ser somados para formar o próximo. Na linha seguinte haverão N inteiros positivos menores que 1000, indicando os primeiros termos da sequência. Por fim, na última linha há um inteiro I.

Restrições

- $1 \leq N \leq 100$.
- $1 \leq I \leq 10^{18}$.

Saída

Você deve imprimir o I-ésimo termo da sequência mod 1000000009. A indexação começa em 1.

Exemplo de Entrada	**Exemplo de Saída**
2 1 1 7	13
2 1 1 1000000000000000000	209762233