

Competidor(a): _____

Número de inscrição: _____ – _____ (opcional)



OBI2021

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 3

25 de setembro de 2021

A PROVA TEM DURAÇÃO DE 3 HORAS

Promoção:



Sociedade Brasileira de Computação

Apoio:



Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 9 páginas (não contando a folha de rosto), numeradas de 1 a 9. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa; em particular, seu programa não deve escrever frases como “Digite o dado de entrada:” ou similares.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão necessariamente ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou pen-drive, conforme especificado pelo seu professor.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
 - em Pascal: *readln, read, writeln, write*;
 - em C: *scanf, getchar, printf, putchar*;
 - em C++: as mesmas de C ou os objetos *cout* e *cin*.
 - em Java: qualquer classe ou função padrão, como por exemplo *Scanner, BufferedReader, BufferedWriter* e *System.out.println*
 - em Python: *read, readline, readlines, input, print, write*
 - em Javascript: *scanf, printf*
- Procure resolver a tarefa de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Ogro

Nome do arquivo: “ogro.x”, onde x deve ser c, cpp, pas, java, js ou py

O Ogro da Nlogônia está aprendendo a contar até dez usando os dedos das mãos (assim como os humanos, ele possui 2 mãos com 5 dedos cada)). Ele está treinando muito, mas gostaria de ter um aplicativo para ajudá-lo nessa empreitada.

O Ogro aprendeu a mostrar a representação de um número com as mãos da seguinte forma:

- se o número pode ser representado usando apenas uma das mãos, o Ogro usa os dedos na mão esquerda e mantém a mão direita fechada.
- caso contrário, o Ogro mostra todos os cinco dedos da mão esquerda, e na mão direita mostra os dedos que faltam para representar o número.

Por exemplo, para o número 3, o Ogro mostra:

III *

onde cada letra I representa um dedo e a mão fechada é representada pelo símbolo ‘*’ (asterisco). Para o número 8 o Ogro mostra:

IIIII III

Sua tarefa é ajudar o Ogro em seu treinamento, escrevendo um programa para, dado um número entre 0 e 10, mostrar a configuração de dedos correspondente a esse número, de acordo com as regras acima.

Entrada

A primeira e única linha da entrada contém um inteiro N , o número que deve ser representado com os dedos das mãos.

Saída

Seu programa deve produzir duas linhas na saída. A primeira linha deve conter a representação dos dedos da mão esquerda, a segunda linha deve conter a representação dos dedos da mão direita. A letra ‘I’ deve ser usada para representar um dedo, e o caractere ‘*’ (asterisco) deve ser usado para representar a mão fechada (isto é, nenhum dedo mostrado).

Restrições

- $0 \leq N \leq 10$

Exemplo de entrada 1	Exemplo de saída 1
8	IIIII III

Explicação do exemplo 1: para representar o número 8 o Ogro mostra os cinco dedos da mão esquerda e três dedos da mão direita.

Exemplo de entrada 2	Exemplo de saída 2
3	III *

Explicação do exemplo 2: para representar o número 3 o Ogro mostra três dedos da mão esquerda e nenhum dedo na mão direita.

Exemplo de entrada 3	Exemplo de saída 3
0	* *

Explicação do exemplo 3: para representar o número zero o Ogro não mostra nenhum dedo da mão esquerda ou da mão direita.

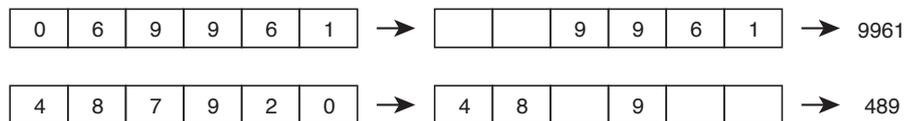
Casamento de inteiros

Nome do arquivo: “casamento.x”, onde x deve ser `c`, `cpp`, `pas`, `java`, `js` ou `py`

Vamos definir a operação de *casamento* de dois números inteiros A e B da seguinte forma:

- inicialmente fazemos A e B terem o mesmo número de dígitos, adicionando zeros à esquerda conforme necessário;
- então cada dígito de A (do menos significativo ao mais significativo) é comparado com o dígito correspondente de B , e o dígito de menor valor é eliminado do número a que pertence (se os dígitos são iguais nenhum é eliminado).
- o resultado da operação de casamento é o par de números inteiros formados pelos dígitos remanescentes de A e B . No caso de não haver dígito remanescente para um dos números, o resultado para esse número é -1 .

Por exemplo, considere o casamento de 69961 com 487920:



O resultado do casamento é o par de números 489 e 9961.

Dados dois números inteiros, sua tarefa é determinar o resultado do casamento desses dois números.

Entrada

A primeira linha da entrada contém um número inteiro A , a segunda linha contém um número inteiro B .

Saída

Seu programa deve produzir uma única linha, contendo os dois números inteiros produzidos pelo casamento dos números dados, em ordem não decrescente.

Restrições

- $1 \leq A \leq 10^9$
- $1 \leq B \leq 10^9$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 22 pontos, $100 \leq A \leq 999$ e $100 \leq B \leq 999$.
- Para um conjunto de casos de testes valendo outros 78 pontos, nenhuma restrição adicional.

Exemplo de entrada 1 69961 487920	Exemplo de saída 1 489 9961
--	---

Explicação do exemplo 1: este exemplo corresponde ao exemplo mostrado no enunciado.

Exemplo de entrada 2	Exemplo de saída 2
5678 1234	-1 5678

Explicação do exemplo 2: todos os dígitos são eliminados do segundo número.

Exemplo de entrada 3	Exemplo de saída 3
21 12	2 2

Explicação do exemplo 3: o dígito 1 é eliminado dos dois números.

Exemplo de entrada 4	Exemplo de saída 4
200 100	0 200

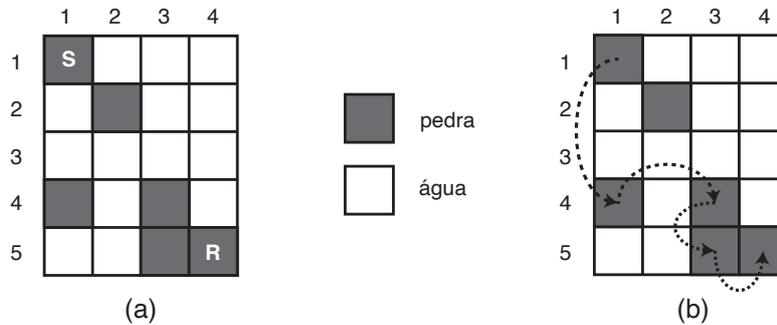
Explicação do exemplo 4: o dígito 1 é eliminado do segundo número.

Sr. Sapo

Nome do arquivo: “sapo.x”, onde x deve ser c, cpp, pas, java, js ou py

O Sr. Sapo mora num lago de formato retangular dividido em um reticulado de células quadradas de um metro de lado. Algumas das células são pedras que estão acima do nível da água.

O Sr. Sapo é muito atlético e pode saltar a distâncias de até três metros, mas curiosamente ele só pode saltar nas direções paralelas aos lados do lago. A figura (a) abaixo mostra um lago, e a figura (b) uma sequência de pulos do Sr. Sapo.



O Sr. Sapo está em uma pedra e quer ir visitar sua namorada que está em outra pedra. Ele está com pressa e não quer se molhar, portanto quer chegar ao seu destino pulando de pedra em pedra, sem cair na água.

Dados o mapa do lago, a pedra em que o Sr. Sapo está e a pedra em que a sua namorada está, determine se é possível ele chegar ao seu destino sem se molhar.

Entrada

A primeira contém dois inteiros N , M , respectivamente a largura e o comprimento do lago em metros (ou seja, o lago é composto por N colunas e M linhas de células quadradas de 1m de lado). As colunas são numeradas de 1 a N e as linhas são numeradas de 1 a M . A segunda linha contém um inteiro P , o número de células que são pedras. Cada uma das P linhas seguintes contém dois inteiros C_i e L_i , respectivamente o número da coluna e o número da linha de uma célula que é pedra. A linha seguinte descreve a célula em que o Sr. Sapo está e contém dois inteiros S_C e S_L , respectivamente a coluna e a linha da célula. A linha seguinte descreve a célula em que está a namorada do Sr. Sapo e contém dois inteiros R_C e R_L , respectivamente a coluna e a linha da célula.

Saída

Seu programa deve produzir uma única linha na saída, contendo um único caractere, que deve ser ‘S’ se for possível o Sr. Sapo chegar ao destino sem se molhar, ou ‘N’ caso contrário.

Restrições

- $3 \leq N \leq 100$
- $1 \leq M \leq 100$
- $2 \leq P \leq N \times M$
- $1 \leq C_i \leq M$ e $1 \leq L_i \leq M$ para $1 \leq i \leq P$
- $1 \leq S_C \leq N$ e $1 \leq S_L \leq M$
- $1 \leq R_C \leq N$ e $1 \leq R_L \leq M$
- As posições do Sr. Sapo e da sua namorada são distintas e ambas são posições de pedras especificadas na entrada.

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 14 pontos, $M = 1$
- Para um conjunto de casos de testes valendo outros 16 pontos, para a pedra em que o Sr. Sapo está inicialmente, há no máximo uma outra pedra para a qual ele pode saltar, e para todas as outras pedras, há no máximo duas para a qual ele pode saltar (ou seja, se o Sr. Sapo consegue chegar ao destino, há um único caminho de pedras que podem ser usadas, e esse caminho não tem "bifurcações").
- Para um conjunto de casos de testes valendo outros 70 pontos, nenhuma restrição adicional.

Exemplo de entrada 1	Exemplo de saída 1
4 5 6 1 1 2 2 1 4 3 4 3 5 4 5 1 1 4 5	S

Explicação do exemplo 1: este exemplo corresponde à figura do enunciado. O Sr. Sapo pode usar as pedras $(1, 1) \rightarrow (1, 4) \rightarrow (3, 4) \rightarrow (3, 5) \rightarrow (4, 5)$ e assim consegue chegar ao destino.

Exemplo de entrada 2	Exemplo de saída 2
4 5 6 2 1 2 5 3 4 4 1 4 3 4 5 3 4 2 1	N

Explicação do exemplo 2: este exemplo corresponde à seguinte figura:

	1	2	3	4
1		R		
2				
3				
4			S	
5				

O Sr. Sapo não consegue dar nenhum pulo e não consegue chegar ao seu destino.

Plano de estacionamento

Nome do arquivo: “plano.x”, onde x deve ser c, cpp, pas, java, js ou py

Tio Chico é o dono de um estacionamento para carros, localizado perto de um estádio de futebol. O estacionamento tem N vagas numeradas de 1 a N e em dias de jogo tem muita procura, podendo até mesmo lotar.

Tio Chico é um tanto excêntrico, e decidiu que, no próximo jogo, deverá ser obedecida uma nova regra, que em termos gerais consiste no seguinte: o carro do i -ésimo cliente a chegar deverá ocupar uma vaga cujo número está dentro de um certo intervalo. Esses intervalos foram definidos pelo Tio Chico de acordo com alguns critérios, como espaços para manobra, sombreamento, etc.

Mais especificamente, para o i -ésimo cliente que chegar, Tio Chico definiu um número V_i e determinou que o automóvel desse cliente deve ocupar uma vaga ainda não ocupada cujo número está dentro do intervalo $1, 2, \dots, V_i$. Vamos chamar de *plano de estacionamento* a lista dos valores V_i , para todos os clientes i .

Se um cliente chegar e não puder estacionar o carro de acordo com o plano de estacionamento, esse cliente não será atendido, e o estacionamento não aceitará o carro de nenhum outro cliente até o final do jogo.

Você ficou muito preocupado com essa esquisitice do Tio Chico, e conhecendo o plano de estacionamento que foi definido, precisa determinar qual o maior número de clientes que poderão estacionar.

Entrada

A primeira linha da entrada contém um inteiro N , o número de vagas do estacionamento. A segunda linha contém um inteiro M , o número esperado de clientes. Cada uma das M linhas seguintes contém um inteiro V_i , o número definido no plano de estacionamento para o i -ésimo cliente a chegar.

Saída

Se programa deve produzir uma única linha, contendo um único inteiro, o número máximo de carros que poderão estacionar de acordo com o plano de estacionamento de Tio Chico.

Restrições

- $1 \leq N \leq 100\ 000$
- $1 \leq M \leq 100\ 000$
- $1 \leq V_i \leq N$, para $1 \leq i \leq N$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 40 pontos, $N \leq 2000$ e $M \leq 2000$.
- Para um conjunto de casos de testes valendo outros 60 pontos, nenhuma restrição adicional.

Exemplo de entrada 1	Exemplo de saída 1
4	2
3	
4	
1	
1	

Explicação do exemplo 1: O carro do cliente 1 pode estacionar em qualquer vaga do estacionamento, mas é melhor não ocupar a vaga 1. O carro do cliente 2 então ocupa a vaga 1. O carro do cliente 3 não pode estacionar, porque a vaga 1 já está ocupada.

Exemplo de entrada 2	Exemplo de saída 2
4 6 2 2 3 3 4 4	3

Explicação do exemplo 2: Os carros dos dois primeiros clientes ocupam as vagas 1 e 2, em qualquer ordem. O carro do cliente 3 ocupa a vaga 3. Então o carro do cliente 4 não pode estacionar pois todas as vagas de 1 a 3 estão ocupadas, e a resposta é 3.