Competidor(a): _____

Número de inscrição: _____–_____ *(opcional)*

Este Caderno de Tarefas não pode ser levado para casa após a prova. Após a prova entregue este Caderno de Tarefas para seu professor guardar. Os professores poderão devolver os Cadernos de Tarefas aos competidores após o término do período de aplicação das provas (1 de outubro de 2022).

Olimpíada Brasileira de Informática

OBI2022

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 3

1 de outubro de 2022

A PROVA TEM DURAÇÃO DE 3 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

FUNDAÇÃO BEHRING

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 8 páginas (não contando a folha de rosto), numeradas de 1 a 8. 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.

Jogo

Nome do arquivo: `jogo.c`, `jogo.cpp`, `jogo.pas`, `jogo.java`, `jogo.js` *ou* `jogo.py`

Uma empresa está desenvolvendo um aplicativo para celular que tem como objetivo estimular o gosto por matemática em jovens.

O aplicativo é um jogo, chamado Maior ou Menor, que sorteia um número inteiro e o jogador tem que adivinhar qual o número sorteado. O jogo é composto de uma ou mais rodadas. A cada rodada, o jogador digita um número e o aplicativo responde com:

- `menor` se o número digitado é maior do que o sorteado;
- `maior` se o número digitado é menor do que o sorteado; e
- `correto` se o número digitado é igual ao número sorteado.

O jogo termina quando o jogador acerta o número sorteado.

Dados o número sorteado e as tentativas de um jogador, você deve escrever um programa que simule o comportamento do aplicativo.

Entrada

A primeira linha contém um inteiro X, o número sorteado pelo aplicativo. Cada uma das linhas seguintes contém um inteiro T, indicando uma tentativa do jogador de acertar o número sorteado. Na última linha da entrada, $T = X$.

Saída

Para cada tentativa do jogador seu programa deve produzir uma linha na saída, contendo apenas uma palavra, que deve ser `menor` se o valor da tentativa é maior do que o número sorteado, `maior` se o valor da tentativa é menor do que o número sorteado ou `correto` se o valor da tentativa é igual ao número sorteado.

Restrições

- $-10\,000 \leq X \leq 10\,000$
- $-10\,000 \leq T \leq 10\,000$
- Haverá no máximo 1000 rodadas em cada jogo.

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 43 pontos, é garantido que o jogo termina em exatamente duas rodadas.
- Para um outro conjunto de casos de testes valendo 57 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
103 1000 1 500 200 100 110 105 103	menor maior menor menor maior menor menor correto

Exemplo de entrada 2	**Exemplo de saída 2**
10000 10000	correto

Exemplo de entrada 3	**Exemplo de saída 3**
-2 0 -1 -2	menor menor correto

Teste de redação

Nome do arquivo: `teste.c`, `teste.cpp`, `teste.pas`, `teste.java`, `teste.js` *ou* `teste.py`

Você é o mais novo estagiário numa empresa de inteligência artificial que está desenvolvendo um corretor automatizado de redações. Para testar o corretor, seu chefe solicitou que você escreva um programa que gere redações aleatórias – que por não fazerem sentido, devem receber a nota zero do corretor automatizado!

As redações geradas pelo seu programa devem obedecer aos seguintes requisitos:

- Cada palavra deve ser composta apenas por letras minúsculas.
- Cada palavra deve ter no mínimo uma letra e no máximo 10 letras.
- Duas palavras devem ser separadas por um espaço em branco.
- A redação deve ser composta por no mínimo N palavras e no máximo M palavras.
- A redação deve ser composta por no mínimo $M/2$ palavras distintas.

Entrada

A primeira linha contém dois inteiros N e M, indicando respectivamente o número mínimo e o número máximo de palavras da redação teste.

Saída

Seu programa deve produzir uma única linha, contendo a redação produzida.

Restrições

- $1 \leq N \leq M \leq 10\,000$
- M é par
- Apenas letras minúsculas não acentuadas são permitidas: `abcdefghijklmnopqrstuvwxyz`

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 33 pontos, $26 \leq M \leq 52$.
- Para outro conjunto de casos de testes valendo 67 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`2 10`	`melhor que machado de assis`

Explicação do exemplo 1: A redação deve ter no mínimo duas e no máximo dez palavas, então uma redação com cinco palavras obedece ao critério do número de palavras. Além disso, a redação contém cinco palavras distintas, o que obedece ao critério de número de palavras distintas (mínimo de cinco). Então a redação produzida está correta.

Exemplo de entrada 2	**Exemplo de saída 2**
`6 6`	`b c d e f g`

Explicação do exemplo 2: A redação deve ter exatamente seis palavras, então a redação produzida tem o número correto de palavras. Além disso, a redação contém seis palavras distintas, mais do que o mínimo exigido (três). Então a redação produzida está correta.

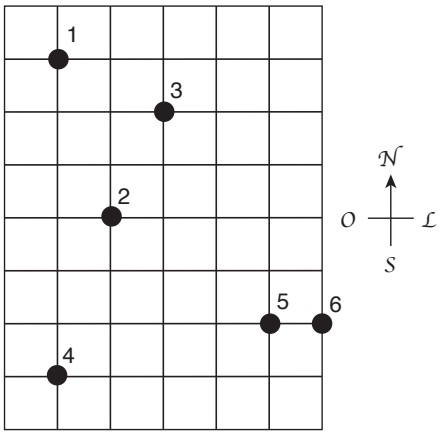
Exemplo de entrada 3	**Exemplo de saída 3**
`6 8`	`aa bb aa bb aa bb`

Explicação do exemplo 3: A redação deve ter no mínimo seis e no máximo oito palavras, então uma redação com seis palavras é correta. No entanto a redação produzida não tem o número mínimo de palavras distintas necessário (tem três, deveria ter no mínimo quatro), então a redação mostrada está INCORRETA.

Carro elétrico

Nome do arquivo: `carro.c`, `carro.cpp`, `carro.pas`, `carro.java`, `carro.js` *ou* `carro.py`

O mapa ao lado mostra o reino de Quadradônia. As estradas são representadas por linhas e as cidades por círculos numerados de 1 a 10. As estradas são igualmente espaçadas com distância de 100 km entre cada par de estradas, sendo orientadas em apenas duas direções: Norte-Sul e Leste-Oeste. Uma empresa de aluguel de carros em Quadradônia utiliza apenas carros elétricos. A *autonomia* de um carro elétrico é a distância que ele pode percorrer com uma carga de energia; após essa distância o carro deve ser carregado novamente para que possa ser utilizado. Há carregadores de energia em cada cidade e não há carregadores de energia fora das cidades. Entre cidades, os carros trafegam apenas pelas estradas e todos os carros têm a mesma autonomia.



Um vendedor deseja partir da cidade 1 e visitar todas as outras cidades, em qualquer ordem, mesmo que ele visite a mesma cidade mais de uma vez. Ele quer utilizar preferencialmente carros elétricos na sua viagem, mas se necessário viajará de avião se a distância para a próxima cidade for maior do que a autonomia do carro. Por exemplo, no mapa acima, se a autonomia for 300 km, o vendedor pode alugar um carro em 1 e visitar 3 e depois 2, mas não pode alcançar as outras cidades. Então ele pode viajar de avião até 5, alugar um carro visitar 6, depois viajar de avião até 4. Assim, são necessárias duas viagens de avião para ele visitar todas as cidades.

Dados o mapa da Quadradônia e a autonomia dos carros, determine qual o menor número de viagens de avião que são necessárias para que o viajante visite todas as cidades, partindo da cidade 1.

Entrada

A primeira linha contém dois inteiros X e Y, indicando respectivamente o número de estradas na direção Oeste-Leste e estradas na direção Norte-Sul. As estradas são numeradas de 1 a X na direção Oeste-Leste e de 1 a Y na direção Norte-Sul. A segunda linha contém dois inteiros N e A, indicando respectivamente o número de cidades e a autonomia dos carros, em kilômetros. As cidades são numeradas de 1 a N. Cada uma das N linhas seguintes contém um par de inteiros x_i e y_i, indicando a posição da cidade de número i, para $1 \leq i \leq N$.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o menor número de viagens de avião necessárias para que o vendedor visite todas as cidades.

Restrições

- $1 \leq X \leq 100\ 000$
- $1 \leq Y \leq 100\ 000$
- $2 \leq N \leq 1\ 000$
- $1 \leq A \leq 150\ 000$
- $1 \leq x_i \leq X$
- $1 \leq y_i \leq Y$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 37 pontos, $N = 2$.
- Para outro conjunto de casos de testes valendo 32 pontos, $Y = 1$, e é garantido que as cidades são dadas em ordem crescente de X (isto é, $x_1 < x_2 < ... < x_n$).
- Para um outro conjunto de casos de testes valendo 31 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
7 9 6 300 2 2 3 5 4 3 2 8 6 7 7 7	2

Explicação do exemplo 1: Este é o exemplo do enunciado.

Exemplo de entrada 2	**Exemplo de saída 2**
7 9 6 200 2 2 3 5 4 3 2 8 6 7 7 7	4

Explicação do exemplo 2: Como a autonomia é 200 km, a única viagem de carro possível é entre as cidades 5 e 6. O vendedor pode por exemplo viajar de avião de 1 para 3, depois de 3 para 2, depois de 2 para 4, depois de 4 para 5, alugar um carro e visitar 6, para um total de quatro viagens de avião.

Dígitos

Nome do arquivo: `digitos.c`, `digitos.cpp`, `digitos.pas`, `digitos.java`, `digitos.js` *ou* `digitos.py`

Joãozinho te propôs o seguinte desafio: ele escolheu dois inteiros A e B, com $1 \leq A \leq B \leq 10^{1000}$, e escreveu na lousa todos os inteiros entre A e B, em sequência, porém colocando um espaço após cada dígito, de forma a não ser possível ver quando um número termina ou começa. Por exemplo, se Joãozinho escolher $A = 98$ e $B = 102$, ele escreveria a sequência "9 8 9 9 1 0 0 1 0 1 1 0 2".

Seu desafio é: dada a lista de dígitos escritos na lousa, encontrar os valores de A e B. Caso exista mais de uma possibilidade para os valores que geraria a lista, você deve encontrar uma em que o valor de A é o menor possível.

É garantido que a lista de dígitos da lousa tem no máximo tamanho 1000.

Entrada

A primeira linha da entrada contém um único inteiro N, indicando o número de dígitos. A segunda linha contém N inteiros d_i, indicando os dígitos escritos.

Saída

Imprima o menor valor possível de A.

Restrições

- $1 \leq N \leq 1000$
- $0 \leq d_i \leq 9$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 21 pontos, $1000 \leq A \leq B \leq 9999$.
- Para outro conjunto de casos de testes valendo 23 pontos, $B = A + 1$.
- Para outro conjunto de casos de testes valendo 40 pontos, $A, B < 10^6$.
- Para outro conjunto de casos de testes valendo 16 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
6 1 2 3 1 2 4	123

Exemplo de entrada 2	**Exemplo de saída 2**
6 8 9 1 0 1 1	8