OBI2014

Caderno de Tarefas

Modalidade **Universitária**, Fase **1**

10 de maio de 2014

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

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.
- 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 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; e soluções na linguagem Python devem ser arquivos com sufixo *.py*. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou disquete, 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,print,write*
- Procure resolver o problema 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.

Carteiro

Nome do arquivo fonte: `carteiro.c`, `carteiro.cpp`, `carteiro.pas`, `carteiro.java`, *ou* `carteiro.py`

Um carteiro é o responsável por entregar as encomendas na rua de Joãozinho. Por política da empresa, as encomendas devem ser entregues na mesma ordem que foram enviadas, mesmo que essa não seja a forma mais rápida. Cansado de subir e descer aquela rua tantas vezes, nosso amigo quer mostrar à empresa quanto tempo ele leva para entregar as encomendas, na tentativa de derrubar essa política.

A rua de Joãozinho tem N casas. Naturalmente, as casas são numeradas de forma ordenada (não necessariamente por números consecutivos). Como as casas possuem aproximadamente o mesmo tamanho, você pode assumir que o carteiro leva uma unidade de tempo para caminhar de uma casa até a casa imediatamente vizinha.

Há M encomendas para essa rua, que devem ser entregues na mesma ordem em que chegaram. Cada encomenda contém o número da casa onde deve ser entregue.

Escreva um programa que determine quanto tempo o carteiro levará para entregar todas as encomendas, assumindo que quando o tempo começa a contar, ele está na primeira casa (a de menor número), e o tempo termina de contar quando todas as encomendas foram entregues (mesmo que o carteiro não esteja de volta na primeira casa). Você pode desprezar o tempo para colocar a encomenda na caixa de correio (ou seja, se ele só tiver uma encomenda, para a primeira casa, a resposta para o problema é zero).

Entrada

A primeira linha contém dois inteiros, N e M, respectivamente o número de casas e o número de encomendas. A segunda linha contém N inteiros em ordem estritamente crescente, indicando os números das casas. A terceira linha contém M inteiros indicando os números das casas onde as encomendas devem ser entregues, na ordem dada na entrada.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o tempo que o carteiro levará para entregar todas as encomendas na ordem correta, assumindo que ele começa na casa de menor número.

Restrições

- $1 \leq N \leq 45.000$ e $1 \leq M \leq 45.000$
- O número de cada casa é um inteiro entre 1 e 1.000.000.000

Informações sobre a pontuação

- Para um subconjunto dos casos de teste totalizando 30 pontos, $1 \leq N \leq 1000$ e $1 \leq M \leq 1000$.

Exemplos

Entrada	**Saída**
5 5 1 5 10 20 40 10 20 10 40 1	10

Entrada	**Saída**
3 4 50 80 100 80 80 100 50	4

Cartas

Nome do arquivo fonte: `cartas.c`, `cartas.cpp`, `cartas.pas`, `cartas.java`, *ou* `cartas.py`

Beatriz gosta muito de jogar cartas com as amigas. Para treinar memória e raciocínio lógico, ela inventou um pequeno passatempo com cartas. Ela retira as cinco primeiras cartas do topo de um baralho bem embaralhado, e as coloca em sequência, da esquerda para a direita, na mesa, com as faces voltadas para baixo.

Então ela olha, por um breve instante, cada uma das cartas da sequência (e logo as recoloca na mesa, com a face para baixo). Usando apenas a sua memória, Beatriz deve agora dizer se a sequência de cartas está ordenada crescentemente, decrescentemente, ou não está ordenada.

De tanto jogar, ela está ficando cansada, e não confia em seu próprio julgamento para saber se acertou ou errou. Por isso, ela pediu para você fazer um programa que, dada uma sequência de cinco cartas, determine se a sequência dada está ordenada crescentemente, decrescentemente, ou não está ordenada.

Entrada

A entrada consiste de uma única linha que contém as cinco cartas da sequência. Os valores das cartas são representados por inteiros entre 1 e 13. As cinco cartas têm valores distintos.

Saída

Seu programa deve produzir uma única linha, contendo um único caractere maiúsculo: '`C`' caso a sequência dada esteja ordenada crescentemente, '`D`' se estiver ordenada decrescentemente, ou '`N`' caso contrário.

Restrições

- o valor de cada carta é um inteiro entre 1 e 13.

Exemplos

Entrada	**Saída**
`1 2 3 5 6`	`C`

Entrada	**Saída**
`5 7 10 9 11`	`N`

Entrada	**Saída**
`12 10 4 3 2`	`D`

PacMan

Nome do arquivo fonte: `pacman.c`, `pacman.cpp`, `pacman.pas`, `pacman.java`, *ou* `pacman.py`

Pacman é um jogo muito conhecido, onde o personagem tenta comer a maior quantidade possível de bolinhas, tendo ao mesmo tempo que fugir de vários fantasmas. Dessa vez, nosso personagem quer carregar a comida coletada para casa, mas o encontro com um fantasma, ao invés de terminar o jogo, faz com que toda a comida coletada seja roubada.

Neste problema os fantasmas não se movem, e o jogador sempre faz o Pacman percorrer o seguinte caminho:

1. O Pacman começa no canto superior esquerdo do tabuleiro.
2. O Pacman percorre toda a linha, da esquerda para direita, até chegar ao lado direito do tabuleiro.
3. O jogador desce uma posição, e percorre toda a linha, desta vez da direita para a esquerda.
4. As etapas 2 e 3 se repetem até que todo o tabuleiro tenha sido percorrido.

Infelizmente, Pacman não pode ignorar os comandos do usuário para fugir dos fantasmas ou pegar mais comida, mas ele pode, a qualquer momento, se aproveitar de um bug de implementação e interromper o jogo, levando consigo toda a comida que estiver carregando.

Você deve escrever um programa que determine a maior quantidade de comida que o Pacman pode levar, se escolher a melhor hora possível para sair. Note que o jogador também tem a opção de não sair antes do final do jogo.

Entrada

A primeira linha contém um inteiro N, o tamanho do tabuleiro do jogo, que é quadrado. Cada uma das N linhas seguintes contém N caracteres, que podem ser (aspas para melhor clareza):

- '`.`' um espaço vazio;
- '`o`' uma comida;
- '`A`' um fantasma.

Saída

Seu programa deve produzir uma única linha contendo um único inteiro, a quantidade máxima de comida que o Pacman pode levar para casa.

Restrições

- $2 \leq N \leq 100$
- Não há um fantasma e uma comida na mesma posição.
- Não há fantasma nem comida na posição inicial do Pacman (ou seja, o primeiro caractere da primeira linha do tabuleiro é '`.`').

Exemplos

Entrada	**Saída**
5 .ooo. ..ooA ..Aoo Aoooo ..ooo	6

Entrada	**Saída**
3 .o. oAA ooo	4

Fechadura

Nome do arquivo fonte: `fechadura.c`, `fechadura.cpp`, `fechadura.pas`, `fechadura.java`, *ou* `fechadura.py`

Joãozinho estava um dia chegando em casa quando percebeu que havia perdido a chave da porta. Desesperado, ele resolveu pedir ajuda a seu amigo Roberto, que em poucos segundos conseguiu abrir a porta usando suas ferramentas.

Admirado com a velocidade em que seu amigo conseguiu abrir a porta de sua casa sem a chave, ele decidiu perguntar como ele tinha conseguido aquilo. Roberto explicou que a fechadura da casa de Joãozinho é baseada em um sistema de pinos de tamanhos diferentes que, uma vez alinhados na mesma altura M, possibilitam a abertura da porta.

Uma fechadura é um conjunto de N pinos dispostos horizontalmente que podem ser movimentados para cima ou para baixo com o auxílio de uma chave de metal que, ao ser inserida dentro da fechadura, pode aumentar ou diminuir em 1mm, simultaneamente, a altura de quaisquer dois pinos consecutivos.

Joãozinho como um exemplar perfeccionista decidiu desbloquear sua fechadura na menor quantidade de movimentos, onde cada movimento consiste em escolher dois pinos consecutivos da fechadura e aumentar ou diminuir a altura dos dois pinos em 1mm. Após todos os pinos possuírem altura exatamente igual a M, a fechadura é desbloqueada.

Entrada

A primeira linha da entrada contém dois inteiros N e M representando, respectivamente, a quantidade de pinos da fechadura e a altura em que eles devem ficar para a fechadura ser desbloqueada.

A segunda linha da entrada contém N inteiros, representando as alturas dos pinos da fechadura.

Saída

Seu programa deve imprimir uma linha contendo um inteiro representando a quantidade mínima de movimentos para desbloquear a fechadura.

Restrições

- $1 \leq N \leq 1000$
- $1 \leq M \leq 100$
- Cada altura dos pinos está entre 1 e 100.
- É garantido que os casos de testes sempre possuem uma solução.

Exemplos

Entrada	**Saída**
`4 50` `45 45 55 55`	`10`

Entrada	**Saída**
`5 84` `84 39 17 72 94`	`77`

Matriz Escada

Nome do arquivo fonte: `escada.c`, `escada.cpp`, `escada.pas`, `escada.java`, *ou* `escada.py`

Joãozinho está aprendendo sobre matrizes. Hoje ele aprendeu como deixar matrizes na forma escada, e está exercitando. Para ajudá-lo, você deve escrever um programa que determine se o resultado dele realmente está no formato correto.

Uma matriz está na forma escada quando, para cada linha, as condições a seguir forem satisfeitas:

- Se a linha só possuir zeros, então todas as linhas abaixo desta também só possuem zeros.
- Caso contrário, seja X o elemento diferente de zero mais à esquerda da linha; então, para todas as linhas abaixo da linha de X, todos os elementos nas colunas à esquerda de X e na coluna de X são iguais a zero.

Entrada

A primeira linha possui dois inteiros N e M, as dimensões da matriz. Cada uma das N linhas seguintes contém M inteiros não-negativos, os elementos da matriz.

Saída

Seu programa deve produzir uma única linha, contendo o caractere '`S`' caso a matriz esteja no formato escada, ou '`N`', caso contrário.

Restrições

- $1 \leq N \leq 500$ e $1 \leq M \leq 500$.
- Cada elemento da matriz está entre 0 e 10^5.

Informações sobre a pontuação

- Em um conjunto de casos de teste equivalente a 80 pontos, $N \leq 50$ e $M \leq 50$.

Exemplos

Entrada	**Saída**
4 6 1 2 9 9 9 9 0 0 3 9 9 9 0 0 0 0 5 9 0 0 0 0 0 6	S

Entrada	**Saída**
5 8 0 5 1 0 3 2 2 0 0 0 0 0 4 0 1 2 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	S

Entrada	**Saída**
5 5 1 1 2 3 4 0 1 1 4 5 0 1 2 3 6 0 0 0 2 0 0 0 0 0 0	N