OBI2014

Caderno de Tarefas

Modalidade **Programação** • Nível **1**, Fase **1**

24 de maio de 2014

A PROVA TEM DURAÇÃO DE **4** 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 6 páginas (não contando a folha de rosto), numeradas de 1 a 6. 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.

Triângulo

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

Ana e suas amigas estão fazendo um trabalho de geometria para o colégio, em que precisam formar vários triângulos, numa cartolina, com algumas varetas de comprimentos diferentes. Logo elas perceberam que não dá para formar triângulos com três varetas de comprimentos quaisquer. Se uma das varetas for muito grande em relação às outras duas, não dá para formar o triângulo. Ana fez uma pesquisa na internet e aprendeu que com três varetas é possível formar um triângulo quando, para todas as varetas, vale a seguinte relação: o comprimento da vareta é **menor** do que a soma dos comprimentos das outras duas varetas. Por exemplo, se os comprimentos forem 6, 9 e 5, vai dar para formar o triângulo, pois a relação vale para as três varetas: $6 < 9 + 5$, $9 < 6 + 5$ e $5 < 6 + 9$. Mas, se os comprimentos forem, por exemplo, 4, 10 e 3, não vai dar para formar um triângulo, porque a relação não vale para uma das varetas (pois 10 não é menor do que $3 + 4$).

Neste problema, você precisa ajudar Ana e suas amigas a descobrir se, dados os comprimentos de quatro varetas, é ou não é possível selecionar três varetas, dentre as quatro, e formar um triângulo!

Entrada

A entrada é composta por apenas uma linha contendo quatro números inteiros.

Saída

Seu programa deve produzir apenas uma linha contendo o caractere '`S`', caso seja possível formar o triângulo; ou o caractere '`N`', caso não seja possível formar o triângulo.

Restrições

- O valor dos quatro números está entre 1 e 100.

Exemplos

Entrada	**Saída**
`6 9 22 5`	`S`

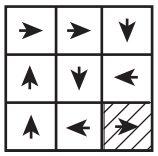
Entrada	**Saída**
`14 40 12 60`	`N`

Setas

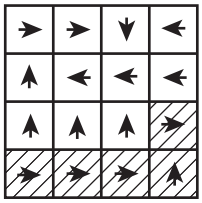
Nome do arquivo fonte: `setas.c`, `setas.cpp`, `setas.pas`, `setas.java`, *ou* `setas.py`

Gabriel é um garoto que gosta muito de um jogo onde há várias letras em um tabuleiro e o jogador precisa rapidamente pisar nas letras corretas, de acordo com as instruções na tela, seguindo uma música. Cansado de vencer, Gabriel inventou um novo jogo: agora temos um tabuleiro quadrado, com N células de cada lado, em que cada célula possui uma *seta* que aponta para uma das quatro posições vizinhas. O jogador primeiro escolhe uma célula inicial para se posicionar e, quando a música começa, ele deve caminhar na direção para onde a seta em que ele está aponta. Ganha o jogo quem pisar em mais setas corretas durante um período de tempo.

O problema é que Gabriel joga tão rápido que quando a seta atual manda ele sair do tabuleiro, ele segue a orientação, muitas vezes quebrando alguns objetos próximos. Quando isso acontece, dizemos que a célula inicial deste jogo não é segura, pois leva a um caminho que termina fora do tabuleiro. A figura abaixo mostra dois tabuleiros.



Tabuleiro 3x3 com oito células seguras



Tabuleiro 4x4 com onze células seguras

Ajude Gabriel: dada a configuração do tabuleiro, determine quantas células são seguras para ele iniciar o jogo.

Entrada

A primeira linha da entrada contém um inteiro N, o tamanho do tabuleiro. Cada uma das N linhas seguintes contém N caracteres, com as direções das setas. As direções válidas são:

- '`V`' Aponta para a célula da linha abaixo, na mesma coluna
- '`<`' (sinal menor-que) aponta para a célula à esquerda, na mesma linha
- '`>`' (sinal maior-que) aponta para a célula à direita, na mesma linha
- '`A`' Aponta para a célula da linha acima, na mesma coluna

Saída

Seu programa deve produzir um único inteiro, o número de células seguras.

Restrições

- $1 \leq N \leq 500$

Informações sobre a pontuação

- Em um subconjunto dos casos totalizando 50 pontos, $1 \leq N \leq 50$.

Exemplos

Entrada	**Saída**
3 >>V AV< A<>	8

Entrada	**Saída**
4 >>V< A<<< AAA> >>>A	11

Semente

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

Um experimento biológico utiliza uma fita de papel branco especial, na qual algumas gotas de um reagente são colocadas em posições específicas. Inicialmente a gota de reagente faz com que o papel se torne preto na posição em que foi colocada. A cada dia o reagente se propaga pelo papel, em todas as direções, com velocidade de 1 posição por dia, colorindo a região em que o reagente se propagou. A figura abaixo mostra um experimento com uma fita de 13 posições, com três gotas de reagente inicialmente, colocadas nas posições 2, 6 e 13 (a posição 1 é a primeira mais à esquerda da fita). Ao final do terceiro dia, a fita está completamente tomada pelo reagente.

Fita no estado inicial

Fita após um dia

Fita após dois dias

Fita após três dias

Você foi contratado para escrever um programa que, dados o comprimento da fita de papel e as posições das gotas de reagente no início do experimento, determine quantos dias serão necessários para a fita de papel ficar completamente tomada pelo reagente.

Entrada

A primeira linha contém dois inteiros F e R, indicando respectivamente o comprimento da fita de papel, em números de posições, e o número de gotas no início do experimento. A segunda linha contém R inteiros, indicando as posições das gotas de reagente, que são dadas em ordem crescente.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de dias necessários para que a fita de papel fique totalmente tomada pelo reagente.

Restrições

- $1 \leq F \leq 100000$, $1 \leq R \leq 1000$

Informações sobre a pontuação

- Em um conjunto de casos de teste equivalente a 80 pontos, $F \leq 1000$.

Exemplos

Entrada	**Saída**
13 3 2 6 13	3

Entrada	**Saída**
10 2 9 10	8

Letras

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

Considere as definições abaixo:

- Uma *palavra* é uma sequência de letras consecutivas.
- Um *texto* é um conjunto de palavras separadas pelo caractere *espaço em branco*.

Você foi contratado pela empresa Booble para escrever um programa que, dados uma letra e um texto, determina a porcentagem de palavras do texto que contém a letra dada.

Entrada

A primeira linha da entrada contém um único caractere, a letra de interesse na pesquisa. A segunda linha contém um texto, como definido acima.

Saída

Seu programa deve produzir uma única linha, contendo um único número real, a porcentagem de palavras do texto que contêm a letra dada, com precisão de uma casa decimal.

Restrições

- O texto é composto apenas por letras minúsculas e o caractere espaço em branco.
- O texto é formado por no mínimo um caractere, e no máximo 1000 caracteres.
- O texto não contém dois espaços em branco consecutivos.

Exemplos

Entrada	**Saída**
`p` `papagaio`	`100.0`

Entrada	**Saída**
`o` `no meio do caminho tinha uma pedra tinha uma pedra no meio do caminho`	`57.1`

Entrada	**Saída**
`b` `nunca me esquecerei que no meio do caminho tinha uma pedra`	`0.0`