OBI2017

Caderno de Tarefas

Modalidade **Universitária** • Fase **1**

12 de maio de 2017

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

v1.1

Instruções

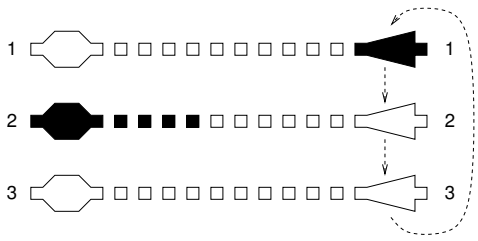
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 4 páginas (não contando a folha de rosto), numeradas de 1 a 4. 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; soluções na linguagem Python devem ser arquivos com sufixo *.py2* para python2 e .py3 para python3; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*. 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 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 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.

Game-10

Nome do arquivo: `game10.c`, `game10.cpp`, `game10.pas`, `game10.java`, `game10.js` *ou* `game10.py`

No princípio dos anos 1980 surgiram nos colégios os primeiros relógios de pulso digitais com joguinhos. Era uma febre entre os alunos e quem tinha um era muito popular na hora do recreio. Os joguinhos eram bem simples, mas muito legais. Um dos primeiros era o Game-10, no qual você controlava um avião que aparecia na parte direita do visor. Na parte esquerda aparecia um disco voador em qualquer uma de três posições, aleatoriamente, e lançava um míssil. O objetivo do jogador era movimentar o avião verticalmente para que ficasse na frente do disco voador (na mesma linha horizontal, do lado direito) e atirar para interceptar o míssil antes que esse atingisse o avião.



Como o movimento do avião era feito com apenas um botão, só dava para movimentar em um sentido: ao apertar o botão sucessivas vezes, o avião se movia na sequência de posições $\cdots 1 \to 2 \to 3 \to 1 \to 2 \to 3 \to 1 \cdots$ Veja que, na situação da figura, o jogador deveria apertar o botão apenas uma vez, para ir da posição 1 para a posição 2, e conseguir atirar e interceptar o míssil.

Neste problema vamos considerar que existem N posições e não apenas três. Dado o número de posições N, a posição D na qual o disco voador aparece, e a posição A onde está o avião, seu programa deve computar o número mínimo de vezes que o jogador precisa apertar o botão para movimentar o avião até a mesma posição do disco voador e poder atirar!

Entrada

A primeira linha da entrada contém um inteiro N, o número de posições. A segunda linha contém um inteiro D, a posição do disco voador. A terceira linha contém um inteiro A, a posição do avião.

Saída

Seu programa deve imprimir uma linha contendo um inteiro, o número mínimo de vezes que o jogador deve apertar o botão para poder atirar.

Restrições

- $3 \leq N \leq 100$
- $1 \leq D, A \leq N$

Exemplos

Entrada	**Saída**
3 2 1	1

Entrada	**Saída**
20 8 13	15

Palíndromo

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

Uma cadeia de caracteres é chamada de palíndromo se a sequência de caracteres da esquerda para a direita é igual à sequência de caracteres da direita para a esquerda (uma outra definição é que o primeiro caractere da cadeia deve ser igual ao último caractere da cadeia, o segundo caractere deve ser igual ao penúltimo caractere, o terceiro caractere deve ser igual ao antepenúltimo caractere, e assim por diante).

Por exemplo, as cadeias de caracteres 'mim', 'axxa' e 'ananaganana' são exemplos de palíndromos. Se uma cadeia não é palíndroma, ela pode ser dividida em cadeias menores que são palíndromas. Por exemplo, a cadeia 'aaxyx' pode ser dividida de quatro maneiras distintas, todas elas contendo apenas cadeias palíndromas: {'aa', 'xyx'}, {'aa', 'x', 'y', 'x'}, {'a', 'a', 'xyx'} e {'a', 'a', 'x', 'y', 'x'}.

Dada uma cadeia de caracteres, escreva um programa que determine qual o menor número de partes em que a dada cadeia deve ser dividida de forma que todas as partes sejam palíndromos.

Entrada

A primeira linha contém um número inteiro N que indica o número de caracteres da cadeia. A segunda linha contém a cadeia de caracteres, composta por letras minúsculas, de 'a' a 'z', sem espaços em branco.

Saída

Seu programa deve produzir uma única linha, contendo um número inteiro, o menor número de partes em que a cadeia de entrada deve ser dividida de forma que todas as partes sejam palíndromes.

Restrições

- $1 \leq N \leq 2000$
- A cadeia de caracteres contém apenas letras minúsculas não acentuadas ('a' a 'z').

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, $N \leq 8$
- Em um conjunto de casos de teste somando 40 pontos, $N \leq 300$

Exemplos

Entrada	**Saída**
3 axa	1

Entrada	**Saída**
6 xyzyyx	4

Entrada	**Saída**
10 bbabcbbaab	4

Botas Trocadas

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

A divisão de Suprimentos de Botas e Calçados do Exército comprou um grande número de pares de botas de vários tamanhos para seus soldados. No entanto, por uma falha de empacotamento da fábrica contratada, nem todas as caixas entregues continham um par de botas correto, com duas botas do mesmo tamanho, uma para cada pé. O sargento mandou que os recrutas retirassem todas as botas de todas as caixas para reembalá-las, desta vez corretamente.

Quando o sargento descobriu que você sabia programar, ele solicitou com a gentileza habitual que você escrevesse um programa que, dada a lista contendo a descrição de cada bota entregue, determina quantos pares corretos de botas poderão ser formados no total.

Entrada

A primeira linha da entrada contém um inteiro N indicando o número de botas individuais entregues. Cada uma das N linhas seguintes descreve uma bota, contendo um número inteiro M e uma letra L, separados por um espaço em branco. M indica o número do tamanho da bota e L indica o pé da bota: $L =$ '`D`' indica que a bota é para o pé direito, $L =$ '`E`' indica que a bota é para o pé esquerdo.

Saída

Seu programa deve imprimir uma única linha contendo um único número inteiro indicando o número total de pares corretos de botas que podem ser formados.

Restrições

- $2 \leq N \leq 10^4$
- N é par
- $30 \leq M \leq 60$
- L é o caractere '`D`' ou o caractere '`E`'

Exemplos

Entrada	**Saída**
4 40 D 41 E 41 D 40 E	2

Entrada	**Saída**
6 38 E 39 E 40 D 38 D 40 D 37 E	1