OBI2014

Caderno de Tarefas

Modalidade **Programação** • Nível **1**, Fase **2**

30 de agosto 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.

Decifra

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

Dimas é um renomado investigador de roubos a antiguidades e obras de arte, que sempre é chamado para casos intrigantes que necessitam de bastante trabalho mental. Desta vez, o quadro que sumiu de um conhecido museu na França foi a Donalisa, do pintor Leonardo da Silva. Este é um caso bastante especial, visto que o ladrão deixou uma frase escrita na parede, aparentemente criptografada. Que desafio para Dimas! É que ele não tem muito conhecimento nessa área de criptografia. Porém, ele usou de suas excelentes observações e conseguiu perceber que a frase foi escrita através de alguma permutação inversível do alfabeto.

Uma permutação inversível do alfabeto é apenas uma troca entre suas letras, duas a duas. Por exemplo, todo "`a`" será trocado por "`m`" e, portanto, todo "`m`" será trocado por "`a`". Dessa forma, veja que dado um texto original, se aplicarmos a permutação, teremos uma frase criptografada; e se aplicarmos a mesma permutação novamente, teremos o texto original recuperado!

Apesar de parecer fácil, a tradução se tornou uma tarefa difícil, já que a frase é bastante longa. É por isso que Dimas resolveu pedir sua ajuda, um exímio programador, para traduzir a frase criptografada, recuperando o texto original, e resolver o mistério!

Entrada

A primeira linha da entrada contém uma sequência de 26 letras minúsculas distintas, representando a permutação inversível usada na frase criptografada. A permutação é a seguinte: a letra "`a`" é trocada pela primeira letra dessa sequência; a letra "`b`" é trocada pela segunda letra dessa sequência; a letra "`c`" pela terceira; e assim por diante, seguindo a sequência padrão do alfabeto: `abcdefghijklmnopqrstuvwxyz`. A segunda linha da entrada consiste de uma frase criptografada, contendo apenas letras minúsculas.

Saída

Seu programa deve imprimir o texto original, de acordo com a permutação fornecida.

Restrições

- A frase criptografada não excede 10^4 caracteres.

Exemplos

Entrada	**Saída**
`zcbedfghljkinmypqrutsvwxoa` `bzedzeymziluz`	`cadeadonalisa`

Entrada	**Saída**
`iohmunlcawygdfbqpvxzerjskt` `haufhaimihbdqezihib`	`cienciadacomputacao`

Quadrado

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

Um quadrado quase mágico, de dimensões $N \times N$, é um quadrado que obedece à seguinte condição. Existe um número inteiro positivo M tal que: para qualquer linha, a soma dos números da linha é igual a M; e para qualquer coluna, a soma dos números da coluna é também igual a M. O quadrado seria mágico, e não apenas quase mágico, se a soma das diagonais também fosse M. Por exemplo, a figura abaixo, parte (a), apresenta um quadrado quase mágico onde $M = 21$.

4	9	8
11	8	2
6	4	11

(a)

3	6	6
8	6	7
4	3	8

(b)

Laura construiu um quadrado quase mágico e alterou, propositalmente, um dos números! Nesta tarefa, você deve escrever um programa que, dado o quadrado quase mágico alterado por Laura, descubra qual era o número original antes da alteração e qual número foi colocado no lugar. Por exemplo, na parte (b) da figura, o número original era 1, que Laura alterou para 7.

Entrada

A primeira linha da entrada contém apenas um número N, representando a dimensão do quadrado. As N linhas seguintes contêm, cada uma, N números inteiros, definindo o quadrado. A entrada é garantidamente um quadrado quase mágico onde exatamente um número foi alterado.

Saída

Seu programa deve imprimir apenas uma linha contendo dois números: primeiro o número original e depois o número que Laura colocou no seu lugar.

Restrições

- $3 \leq N \leq 50$; e o valor de todos os números está entre 1 e 10000

Exemplos

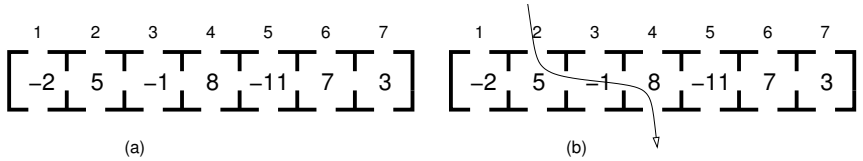
Entrada	**Saída**
3 3 6 6 8 6 7 4 3 8	1 7

Entrada	**Saída**
4 16 3 2 13 5 10 11 8 8 6 7 12 4 15 14 1	9 8

Corredor

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

Bruninho está programando um personagem virtual para o próximo desafio de um jogo de aventura em que, numa das fases, o personagem tem que entrar em um corredor, percorrer algumas salas e depois sair do corredor. Ele pode entrar apenas uma vez, e passar por cada sala apenas uma vez. Todas as salas possuem uma porta de entrada e uma de saída, como ilustra a parte (a) da figura abaixo. Ao passar por uma sala o jogador ganha um certo número de vidas (que pode ser negativo!). O objetivo é passar pelo corredor coletando a maior quantidade possível de vidas! Por sorte, sempre existe ao menos uma sala onde se ganha um número positivo de vidas.



No exemplo acima, o personagem de Bruninho pode ganhar, no máximo, 12 vidas, por exemplo, entrando pela sala 2 e saindo pela sala 4, como mostrado na parte (b) da figura. Nesta tarefa, você deve escrever um programa que, dados os números de vidas correspondentes a cada sala do corredor, calcule a quantidade máxima de vidas que será possível ganhar.

Entrada

A entrada é composta por duas linhas. A primeira linha contém um inteiro N, o número de salas no corredor. A segunda linha contém N números inteiros, positivos ou negativos, indicando a quantidade de vidas que se ganha em cada sala.

Saída

Seu programa deve imprimir uma linha, com o número máximo de vidas que é possível ganhar.

Restrições

- $1 \leq N \leq 50000$; e o número de vidas nas salas está entre -100 e 100

Informações sobre a Pontuação

- Em um conjunto de casos de teste totalizando 30 pontos, $N \leq 1000$

Exemplos

Entrada	**Saída**
7 -2 5 -1 8 -11 7 3	12

Entrada	**Saída**
10 50 42 -35 2 -60 5 30 -1 40 31	105

Jogo da Memória

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

Pedro e Paulo resolveram complicar um pouco o tradicional Jogo da Memória, em que os jogadores precisam virar duas cartas iguais. Eles colocam as cartas no chão, viradas para baixo, e fazem algumas linhas ligando pares de cartas, usando giz, de modo que para qualquer par de cartas (A, B) existe uma e apenas uma sequência de cartas distintas que leva de A até B através das linhas que eles desenharam. Com isso, ao virar duas cartas, o jogador ganha uma quantidade de pontos igual ao tamanho da sequência de linhas entre as duas cartas, se elas forem iguais. Se forem diferentes, o jogador perde aquela quantidade de pontos.

Pedro e Paulo, agora, estão estudando qual é a melhor estratégia para esse jogo e precisam da sua ajuda para resolver uma tarefa específica: dadas as ligações entre as N cartas, calcular a soma dos tamanhos das sequências entre todos os $N/2$ pares de cartas iguais!

O jogo possui N cartas, de índices 1 até N. Cada carta possui a figura de um número de 1 até $N/2$ desenhada. Exatamente duas cartas possuem a figura de cada número entre 1 e $N/2$.

Entrada

A primeira linha da entrada contém o número de cartas N. A segunda linha da entrada contém N inteiros C_i, $1 \leq i \leq N$, indicando qual número está anotado na carta de índice i. Cada uma das $N-1$ linhas seguintes contém dois números A e B, indicando que existe uma linha desenhada entre as cartas de índices A e B.

Saída

Seu programa deve imprimir uma linha contendo um inteiro, a soma dos tamanhos das sequências entre todos os $N/2$ pares de cartas iguais.

Restrições

- $2 \leq N \leq 50000$, N é par
- $1 \leq C_i \leq N/2$
- $1 \leq A \leq N$ e $1 \leq B \leq N$

Informações sobre a Pontuação

- Em um conjunto de casos de teste valendo 50 pontos, $N \leq 1000$

Exemplos

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

Entrada	**Saída**
8 1 2 3 3 2 4 1 4 1 2 2 3 2 6 5 6 6 8 7 8 4 7	12