OBI2015

Caderno de Tarefas

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

29 de maio de 2015

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

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; soluções na linguagem Python devem ser arquivos com sufixo *.py*; 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.

Cobra coral

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

O professor Rui está desenvolvendo um sistema automático para identificar se uma cobra é uma coral verdadeira ou uma falsa coral. A cobra coral verdadeira é venenosa e os anéis coloridos no seu corpo seguem o padrão `...BVBPBVBPBVBP...`, onde `B`,`V` e `P` representam as cores branco, vermelho e preto, respectivamente. Já a falsa coral não é venenosa e os anéis seguem o padrão `...BVPBVPBVPBVP....`

O problema é que os sensores do sistema do professor Rui produzem apenas uma sequência de quatro números representando um pedaço do padrão de cores. Só que ele não sabe qual número representa qual cor. Mas, por exemplo, se a sequência for `5 3 9 3`, podemos dizer com certeza que é uma coral verdadeira, mesmo sem saber qual número representa qual cor! Você deve ajudar o professor Rui e escrever um programa que diga se a coral é verdadeira ou falsa.

Entrada

A entrada consiste de apenas uma linha, contendo quatro números inteiros.

Saída

Seu programa deve imprimir na saída uma linha com a letra "`V`" se a coral for verdadeira ou com a letra "`F`", caso seja falsa.

Restrições

- Os quatro números têm valores entre 1 e 9, inclusive, e a sequência sempre representa uma coral verdadeira, ou uma coral falsa.

Exemplos

Entrada	**Saída**
`5 3 9 3`	`V`

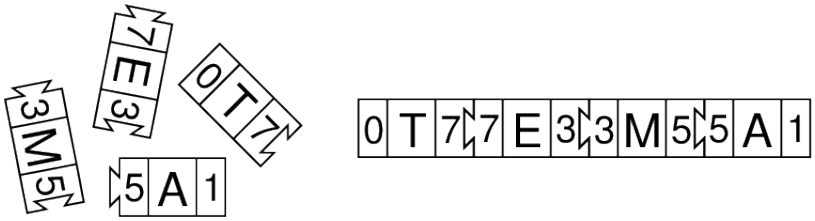
Entrada	**Saída**
`7 1 4 7`	`F`

Entrada	**Saída**
`6 2 6 8`	`V`

Quebra-cabeça

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

Jade precisa da sua ajuda para montar o quebra-cabeças que ela ganhou de presente da sua tia Zoraide! As peças são encaixadas lado a lado e contêm, cada uma, uma letra maiúscula. Quando o quebra-cabeças estiver montado, a sequência de letras revelará uma frase secreta. Cada peça possui, além da letra, dois números: um na parte esquerda e outro na parte direita. Uma peça se encaixa depois de outra, na sequência, quando seu número esquerdo for igual ao número direito da outra peça. O número esquerdo da primeira peça é sempre o 0 (zero) e o número direito da última peça é sempre o 1 (um). Cada número aparece no máximo uma vez na parte esquerda de alguma peça, e no máximo uma vez na parte direita. Sempre é possível encaixar todas as peças e em apenas uma única sequência! Veja um exemplo na figura, com quatro peças formando a palavra "TEMA".



Entrada

A primeira linha da entrada contém um número natural N, indicando o número de peças do quebra-cabeças. As N linhas seguintes contêm, cada uma, a descrição de uma peça na forma E C D, onde: E é o número esquerdo; C é a letra maiúscula; e D é o número direito.

Saída

Seu programa deve escrever uma única linha na saída, contendo a sequência de letras formada quando o quebra-cabeças está montado.

Restrições

- $3 \leq N \leq 100000$; $0 \leq E \leq 200000$; e $0 \leq D \leq 200000$
- Há exatamente uma maneira de montar o quebra-cabeças utilizando todas as peças dadas.

Exemplos

Entrada	**Saída**
4 5 A 1 0 T 7 3 M 5 7 E 3	TEMA

Entrada	**Saída**
3 197452 I 1 0 O 39999 39999 B 197452	OBI

Família real

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

O rei de um reino muito muito distante deu uma grande festa para reunir todas as gerações dos seus descendentes: filhos e filhas, netos e netas, bisnetos e bisnetas, e assim por diante. Ele, que gosta muito de estatísticas, agora quer saber, para cada geração, qual a porcentagem de descendentes daquela geração que compareceu à festa. Você foi contratado para escrever um programa de computador que calcule as porcentagens de todas as gerações!

O rei tem N descendentes, identificados com os números de 1 a N. O próprio rei será identificado com o número 0. Será dada apenas a informação, para cada descendente, de quem é o seu pai ou sua mãe, na linha de descendência que começa no rei. Além disso, claro, será dada a lista de todos que compareceram à festa.

Entrada

A primeira linha da entrada contém dois inteiros N e M, respectivamente, o número de descendentes e o número de participantes da festa. A segunda linha contém N números, representando os pais ou mães dos N descendentes, em ordem crescente: o primeiro número indica o pai ou a mãe do descendente de número 1, o segundo número indica o pai ou a mãe do descendente de número 2, e assim por diante. A terceira linha contém M números, identificando todos os descendentes que compareceram à festa.

Saída

Seu programa deve imprimir uma linha com uma lista de números reais, com precisão de duas casas decimais, indicando a porcentagem, para cada geração, dos descendentes daquela geração que compareceram à festa. O primeiro número deve ser a porcentagem dos filhos e filhas, o segundo dos netos e netas, e assim por diante.

Restrições

- $1 \leq M \leq N \leq 10000$

Exemplos

Entrada	**Saída**
9 5 7 3 0 9 0 3 5 6 7 3 2 8 1 9	50.00 33.33 100.00 0.00

Entrada	**Saída**
16 11 15 9 2 8 6 11 0 3 0 8 12 0 9 6 16 12 5 14 9 12 6 2 4 10 3 11 7	100.00 50.00 66.67 50.00 100.00

Caixinha de palitos

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

A caixinha contém N palitos de picolé, que precisam ser divididos entre os amigos Renato, Gustavo e Bruno, para um trabalho escolar. Cada amigo deve ganhar pelo menos 1 (um) palito. O professor vai determinar um número M máximo de palitos que cada um pode ganhar. Nesta tarefa, dados N e M, seu programa deve calcular quantas maneiras distintas existem de se dividir todos os N palitos entre os três amigos. Por exemplo, para $N = 100$: se $M = 15$, então há zero maneiras de se dividir, pois a soma dos números de palitos de Renato, Gustavo e Bruno seria no máximo 45, só que precisa ser sempre N; mas se $M = 34$, aí veja que haveria 6 maneiras distintas:

	Renato	Gustavo	Bruno
1	34	33	33
2	33	34	33
3	33	33	34
4	34	34	32
5	34	32	34
6	32	34	34

Entrada

A entrada é composta por apenas uma linha com dois números naturais N e M, indicando, respectivamente, o número de palitos na caixinha e o número máximo que cada amigo pode ganhar.

Saída

Seu programa deve escrever uma única linha na saída, contendo um único número natural: quantas maneiras distintas existem de se dividir os N palitos entre os três amigos.

Restrições

- $3 \leq N \leq 100000$, $1 \leq M \leq N$;

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, $N \leq 300$
- Em um conjunto de casos de teste somando 50 pontos, $N \leq 30000$

Exemplos

Entrada	**Saída**
`100 34`	`6`

Entrada	**Saída**
`100 15`	`0`

Entrada	**Saída**
`100000 98765`	`4997567718`

O Banco Inteligente

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

Caixas automáticos nos bancos são uma invenção ótima mas, às vezes, a gente precisa de dinheiro trocado e a máquina entrega notas de R$100,00. Outras vezes, a gente quer sacar um valor um pouco maior e por questões de segurança gostaria de receber tudo em notas de R$100,00, mas a máquina entrega um monte de notas de R$20,00. O Banco Inteligente está tentando minimizar esse problema dando aos clientes a possibilidade de escolher o valor das notas na hora do saque. Para isso, eles precisam da sua ajuda para saber, dado o valor S do saque e quantas notas de cada valor a máquina tem, quantas formas distintas existem de entregar o valor S. O banco disponibiliza notas de 2, 5, 10, 20, 50 e 100. Por exemplo, se $S = 22$ e o número de notas de cada valor é $N_2 = 5, N_5 = 4, N_{10} = 3, N_{20} = 10, N_{50} = 0$ e $N_{100} = 10$, então há 4 formas distintas da máquina entregar o valor do saque: $20 + 2$, $10 + 10 + 2$, $10 + 5 + 5 + 2$ e $5 + 5 + 5 + 5 + 2$.

Entrada

A primeira linha da entrada contém um inteiro S, o valor do saque. A segunda linha contém seis inteiros $N_2, N_5, N_{10}, N_{20}, N_{50}$ e N_{100}, respectivamente, o número de notas de valores $2, 5, 10, 20, 50$ e 100, disponíveis na máquina.

Saída

Seu programa deve imprimir um inteiro, o número de formas distintas da máquina entregar o saque.

Restrições

- $0 \leq S \leq 5000$ e $N_i \leq 500$

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, $N \leq 1000$ e $N_i \leq 30$
- Em um conjunto de casos de teste somando 40 pontos, $N \leq 1000$ e $N_i \leq 50$

Exemplos

Entrada	**Saída**
22 5 4 3 10 0 10	4

Entrada	**Saída**
1000 20 20 20 20 20 20	34201