Competidor(a): _____

Número de inscrição: _____–_____ *(opcional)*

Este Caderno de Tarefas não pode ser levado para casa após a prova. Após a prova entregue este Caderno de Tarefas para seu professor guardar. Os professores poderão devolver os Cadernos de Tarefas aos competidores após o término do período de aplicação das provas (1 a 3 de Junho de 2023).

Olimpíada Brasileira de Informática

OBI2023

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 1

1 a 3 de Junho de 2023

A PROVA TEM DURAÇÃO DE 2 horas

Promoção:

Sociedade Brasileira de Computação

Apoio:

FUNDAÇÃO BEHRING

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 7 páginas (não contando a folha de rosto), numeradas de 1 a 7. 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; em particular, seu programa não deve escrever frases como "Digite o dado de entrada:" ou similares.
- 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 necessariamente 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 Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Você pode submeter até 50 soluções para cada tarefa. A pontuação total de cada tarefa é a melhor pontuação entre todas as submissões. Se a tarefa tem sub-tarefas, para cada sub-tarefa é considerada a melhor pontuação entre todas as submissões.
- 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 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 a tarefa 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.

Prêmio

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

Uma ONG (Organização Não Governamental) oferece cursos gratuitos de programação de computadores, dança, música e culinária. Aproveitando a cozinha montada para os cursos de culinária, também vende pães integrais, doces e bolos para ajudar nas despesas.

O diretor da ONG anunciou um incentivo para a venda da produção da cozinha: considerando que cada pão vale 1 ponto, cada doce vale 2 pontos e cada bolo vale 3 pontos, os colaboradores ganharão um prêmio dependendo da soma total dos pontos dos produtos vendidos durante a semana.

Se a soma dos pontos de todos os produtos vendidos na semana for igual ou maior do que 150, cada colaborador recebe um bolo como prêmio; senão, se a soma dos pontos for maior ou igual a 120, cada colaborador recebe um doce como prêmio; senão, se a soma dos pontos for maior ou igual a 100, cada colaborador recebe um pão como prêmio. Se a soma dos pontos for menor do que 100 não há prêmio para os colaboradores.

Sabendo que você fez um curso de programação na ONG, o diretor pediu que você escreva um programa que, dados os números de pães, doces e bolos vendidos na semana, determine qual o prêmio merecido.

Entrada

A primeira linha contém um inteiro P, o número de pães vendidos na semana. A segunda linha contém um inteiro D, o número de doces vendidos na semana. A terceira e última linha contém um inteiro B, o número de bolos vendidos na semana.

Saída

Seu programa deve produzir uma única linha, contendo um único caractere, indicando o prêmio merecido: a letra maiúscula '`P`' para pão, a letra maiúscula '`D`' para doce, a letra maiúscula '`B`' para bolo e a letra maiúscula '`N`' se os colaboradores não merecem prêmio na semana.

Restrições

- $0 \leq P \leq 100$
- $0 \leq D \leq 100$
- $0 \leq B \leq 100$

Informações sobre a pontuação

- A tarefa vale 100 pontos.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
100 10 4	D

Exemplo de entrada 2	**Exemplo de saída 2**
30 45 10	B

Exemplo de entrada 3	**Exemplo de saída 3**
30 10 5	N

Epidemia

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

Uma nova pandemia é sempre possível (e temida), mas a experiência recente mostrou que atualmente a ciência é capaz de desenvolver vacinas eficazes em muito pouco tempo. Outra consequência da pandemia recente é que muito se estudou sobre epidemias em geral, e vários modelos matemáticos foram desenvolvidos.

Neste problema vamos usar um modelo simples de epidemia:

- Quando uma pessoa é infectada, ela infecta outras R pessoas, mas apenas no dia seguinte à sua infecção (R é chamado de *fator reprodutivo* da infecção).
- Ninguém é infectado mais do que uma vez.

Por exemplo, se no dia 0 da epidemia 3 pessoas são infectadas e o fator reprodutivo R é igual a 2, então no dia 1 outras 6 pessoas são infectadas ($3 + 6 = 9$ pessoas no total), no dia 2 outras 12 pessoas são infectadas ($3 + 6 + 12 = 21$ pessoas no total), no dia 3 outras 24 pessoas infectadas ($3 + 6 + 12 + 24 = 45$ pessoas no total), e assim por diante.

Dados o número inicial de pessoas infectadas no dia 0 e o fator reprodutivo R da epidemia, escreva um programa para determinar qual o número de dias necessários para a epidemia infectar P ou mais pessoas no total.

Entrada

A primeira linha contém um inteiro N, o número de pessoas infectadas no dia 0. A segunda linha contém o fator reprodutivo R da infecção. A terceira e última linha contém um inteiro P, o número alvo de pessoas infectadas.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de dias para P ou mais pessoas serem infectadas.

Restrições

- $1 \leq N \leq 1\ 000$
- $1 \leq R \leq 10$
- $1 \leq P \leq 1\ 000\ 000$

Informações sobre a pontuação

- A tarefa vale 100 pontos.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
1 5 156	3

Exemplo de entrada 2	**Exemplo de saída 2**
2 1 11	5

Chinelos

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

Uma comunidade indígena produz chinelos de juta e criou um site para vender a produção online. Os chinelos são de apenas um tipo, mas são produzidos em vários tamanhos.

Você foi contratado(a) para desenvolver um programa de controle de estoque para o site. O estoque pode ser visto como uma tabela com uma única linha, em que cada coluna representa um tamanho, como mostrado na figura (a) abaixo. Na figura, os tamanhos são representados por números de 1 a 5. Assim, a tabela da figura (a) informa que o estoque do chinelo de tamanho 1 é 4 unidades, e o estoque do chinelo de tamanho 4 é 3 unidades.

Quando um chinelo é vendido, o estoque deve ser atualizado. Por exemplo, se um chinelo de tamanho 1 for vendido, o estoque atualizado é mostrado na figura (b). Se o estoque para um tamanho de chinelo tem valor zero, chinelos desse tamanho não podem ser vendidos (por exemplo o chinelo de tamanho 3). Ou seja, a venda não é efetivada.

Dados o estoque inicial e a lista de pedidos de clientes, escreva um programa para determinar quantos chinelos são efetivamente vendidos no total. Cada pedido se refere a um único chinelo. As vendas são processadas sequencialmente, na ordem em que os pedidos foram feitos. Se uma venda não é possível por falta de estoque, o pedido correspondente é ignorado.

Entrada

A primeira linha da entrada contém um inteiro N, o número de tamanhos de chinelos no estoque. Tamanhos são identificados por inteiros de 1 a N. Cada uma das N linhas seguintes contém N inteiros X_i, indicando a quantidade de chinelos de tamanho i, para $1 \leq i \leq N$. A seguir a entrada contém uma linha com um número inteiro P, o número de pedidos recebidos pela loja. Cada uma das P linhas seguintes contém um inteiro I representando o tamanho do chinelo de um pedido. Os pedidos são dados na ordem em que foram feitos.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número total de chinelos efetivamente vendidos.

Restrições

- $1 \leq N \leq 500$
- $0 \leq X_i \leq 20$ para $1 \leq i \leq N$
- $1 \leq P \leq 1\,000$
- $1 \leq I \leq N$

Informações sobre a pontuação

- A tarefa vale 100 pontos.
- Para um conjunto de casos de testes valendo 27 pontos, $N \leq 3$.
- Para um conjunto de casos de testes valendo outros 73 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 4 2 0 3 2 2 1 3	1

Exemplo de entrada 2	**Exemplo de saída 2**
4 1 3 2 5 4 3 3 3 4	3