Competidor(a): _____

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

OBI2019

Caderno de Tarefas

Modalidade **Programação** • **Nível Júnior** • Fase **Nacional**

21 de setembro de 2019

A PROVA TEM DURAÇÃO DE **3** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 5 páginas (não contando a folha de rosto), numeradas de 1 a 5. 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 tarefas 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 2 devem ser arquivos com sufixo *_py2.py*; soluções na linguagem Python 3 devem ser arquivos com sufixo *_py3.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).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- 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 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.

Pares de números

Nome do arquivo: `pares.c`, `pares.cpp`, `pares.pas`, `pares.java`, `pares.js`, `pares_py2.py` *ou* `pares_py3.py`

Temos um vetor de N inteiros distintos e dois inteiros I e F. Precisamos computar quantos pares desses inteiros do vetor somam pelo menos I e no máximo F. Por exemplo, se o vetor for $[45, 12, 11, 7, 83, 29, 5]$ e $I = 19$ e $F = 52$, temos exatamente 8 pares cuja soma está entre 19 e 52: $\{5, 29\}$, $\{5, 45\}$, $\{7, 12\}$, $\{7, 29\}$, $\{7, 45\}$, $\{11, 12\}$, $\{11, 29\}$ e $\{12, 29\}$.

Entrada

A primeira linha da entrada contém três inteiros N, I e F, indicando respectivamente o tamanho do vetor e o valor mínimo da soma e o valor máximo da soma.

Saída

Seu programa deve imprimir uma única linha contendo um inteiro indicando quantos pares de inteiros no vetor somam pelo menos I e no máximo F.

Restrições

- $2 \leq N \leq 1000$
- $-2000 \leq I, F \leq 2000$
- O valor de cada inteiro no vetor está entre -1000 e 1000
- Os inteiros no vetor são distintos

Exemplo de entrada 1	**Exemplo de saída 1**
`7 19 52` `45 12 11 7 83 29 5`	`8`

Exemplo de entrada 2	**Exemplo de saída 2**
`2 -2 2` `12 -16`	`0`

Parcelamento sem juros

Nome do arquivo: `parcelamento.c`, `parcelamento.cpp`, `parcelamento.pas`, `parcelamento.java`, `parcelamento.js`, `parcelamento_py2.py` *ou* `parcelamento_py3.py`

Pedrinho está implementando o sistema de controle de pagamentos parcelados de uma grande empresa de cartão de crédito digital. Os clientes podem parcelar as compras sem juros no cartão, em até 18 vezes. Quando o valor V da compra é divisível pelo número P de parcelas que o cliente escolhe, todas as parcelas terão o mesmo valor. Por exemplo, se o cliente comprar um livro de $V = 30$ reais em $P = 6$ vezes, então as parcelas terão valores: 5, 5, 5, 5, 5 e 5. Mas se o valor da compra não for divisível pelo número de parcelas será preciso fazer um ajuste, pois a empresa quer que todas as parcelas tenham sempre um valor inteiro e somem no total, claro, o valor exato da compra. O que Pedrinho decidiu foi distribuir o resto da divisão de V por P igualmente entre as parcelas iniciais. Por exemplo, se a compra for de $V = 45$ e o número de parcelas for $P = 7$, então as parcelas terão valores: 7, 7, 7, 6, 6, 6 e 6. Quer dizer, como o resto da divisão de 45 por 7 é 3, então as 3 parcelas iniciais devem ter valor um real maior do que as 4 parcelas finais.

Você precisa ajudar Pedrinho e escrever um programa que, dado o valor da compra e o número de parcelas, imprima os valores de cada parcela.

Entrada

A primeira linha da entrada contém um inteiro V, representando o valor da compra. A segunda linha da entrada contém um inteiro P, indicando o número de parcelas.

Saída

Seu programa deve imprimir P linhas, cada uma contendo um inteiro representando o valor de uma parcela. A i-ésima linha deve conter o valor da i-ésima parcela, para $1 \leq i \leq P$, de acordo com o que Pedrinho decidiu.

Restrições

- $10 \leq V \leq 1000$
- $2 \leq P \leq 18$

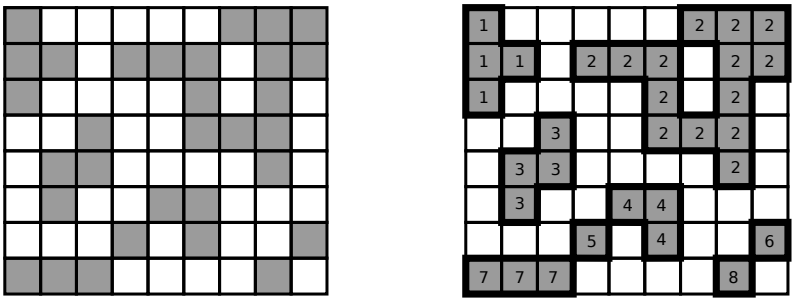
Exemplo de entrada 1	**Exemplo de saída 1**
30 6	5 5 5 5 5 5

Exemplo de entrada 2	**Exemplo de saída 2**
45 7	7 7 7 6 6 6 6

Manchas de pele

Nome do arquivo: `manchas.c`, `manchas.cpp`, `manchas.pas`, `manchas.java`, `manchas.js`, `manchas_py2.py` *ou* `manchas_py3.py`

O laboratório de dermatologia da Linearlândia está implementando um software para contar o número de manchas presentes numa imagem digital de N por M pixels. Cada pixel na imagem é preto ou branco e dois pixels pretos distintos A e B pertencem à mesma mancha se e somente se: existir uma sequência de pixels $[P_1, P_2, \ldots, P_k]$, onde $k \geq 2$, $A = P_1$, $B = P_k$ e para todo $1 \leq i < k$, P_i é ortogonalmente adjacente a P_{i+1} (P_i imediatamente acima, abaixo, à esquerda ou à direita de P_{i+1}).



A figura acima, para $N = 8$ e $M = 9$, ilustra uma imagem digital onde existem oito manchas. Dada a imagem, seu programa deve contar o número de manchas presentes.

Entrada

A primeira linha da entrada contém dois inteiros N e M, representando, respectivamente, o número de linhas e colunas da imagem. As N linhas seguintes contêm, cada uma, M inteiros P representando os pixels da imagem.

Saída

Seu programa deve imprimir uma linha contendo um inteiro, o número de manchas na imagem.

Restrições

- $1 \leq N \leq 1000$
- $1 \leq M \leq 1000$
- O valor de P é 1, representando um pixel preto, ou 0, representando um pixel branco.

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 10 pontos, $N = M = 2$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $N = 1$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $N, M \leq 100$.
- Para um conjunto de casos de testes valendo outros 50 pontos, nenhuma restrição adicional (*Atenção, para essa parcial, não é recomendada uma implementação recursiva!*)

Exemplo de entrada 1	**Exemplo de saída 1**
8 9 1 0 0 0 0 0 1 1 1 1 1 0 1 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 1 1 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 0 0 0 1 0	8

Exemplo de entrada 2	**Exemplo de saída 2**
1 1 0	0

Exemplo de entrada 3	**Exemplo de saída 3**
1 10 0 0 1 0 1 1 1 0 1 0	3