Competidor(a): _____

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

OBI2018

Caderno de Tarefas

Modalidade **Programação** • **Nível 1** • Fase **Local**

17 de maio de 2018

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Instruções

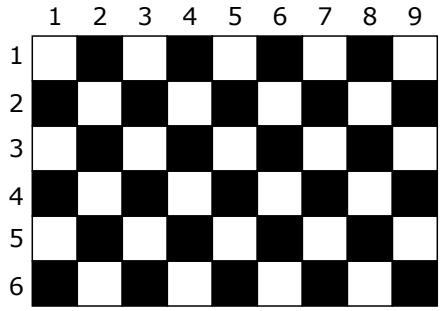
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- 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; 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 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*; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py3*; 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.

Xadrez

Nome do arquivo: "`xadrez.x`", *onde* `x` *deve ser* `cpp`, `pas`, `java`, `js`, `py2` *ou* `py3`

No tabuleiro de xadrez, a casa na linha 1, coluna 1 (canto superior esquerdo) é sempre branca e as cores das casas se alternam entre branca e preta, de acordo com o padrão conhecido como... xadrez! Dessa forma, como o tabuleiro tradicional tem oito linhas e oito colunas, a casa na linha 8, coluna 8 (canto inferior direito) será também branca. Neste problema, entretanto, queremos saber a cor da casa no canto inferior direito de um tabuleiro com dimensões quaisquer: L linhas e C colunas. No exemplo da figura, para $L = 6$ e $C = 9$, a casa no canto inferior direito será preta!



Entrada

A primeira linha da entrada contém um inteiro L indicando o número de linhas do tabuleiro. A segunda linha da entrada contém um inteiro C representando o número de colunas.

Saída

Imprima uma linha na saída. A linha deve conter um inteiro, representando a cor da casa no canto inferior direito do tabuleiro: 1, se for branca; e 0, se for preta.

Restrições

- $1 \leq L, C \leq 1000$

Exemplo de entrada 1	**Exemplo de saída 1**
6 9	0

Exemplo de entrada 2	**Exemplo de saída 2**
8 8	1

Exemplo de entrada 3	**Exemplo de saída 3**
5 91	1

Exemplo de entrada 4	**Exemplo de saída 4**
401 322	0

Escadinha

Nome do arquivo: "`escadinha.x`", *onde* `x` *deve ser* `cpp`, `pas`, `java`, `js`, `py2` *ou* `py3`

Dizemos que uma sequência de números é uma *escadinha*, se a diferença entre números consecutivos é sempre a mesma. Por exemplo, "2, 3, 4, 5" e "10, 7, 4" são escadinhas. Note que qualquer sequência com apenas um ou dois números também é uma escadinha!

Neste problema estamos procurando escadinhas em uma sequência maior de números. Dada uma sequência de números, queremos determinar quantas escadinhas existem. Mas só estamos interessados em escadinhas tão longas quanto possível. Por isso, se uma escadinha é um pedaço de outra, consideramos somente a maior. Por exemplo, na sequência "1, 1, 1, 3, 5, 4, 8, 12" temos 4 escadinhas diferentes: "1, 1, 1", "1, 3, 5", "5, 4" e "4, 8, 12".

Entrada

A primeira linha da entra contém um inteiro N indicando o tamanho da sequência de números. A segunda linha contém N inteiros definindo a sequência.

Saída

Imprima uma linha contendo um inteiro representando quantas escadinhas existem na sequência

Restrições

- $1 \leq N \leq 1000$
- O valor dos números da sequência está entre -10^6 e 10^6 inclusive.

Exemplo de entrada 1	**Exemplo de saída 1**
`8` `1 1 1 3 5 4 8 12`	`4`

Exemplo de entrada 2	**Exemplo de saída 2**
`1` `112`	`1`

Exemplo de entrada 3	**Exemplo de saída 3**
`5` `11 -106 -223 -340 -457`	`1`

Pirâmide

Nome do arquivo: "`piramide.x`", *onde* `x` *deve ser* `cpp`, `pas`, `java`, `js`, `py2` *ou* `py3`

No depósito da fábrica, encostada numa parede, existe uma matriz de N linhas por N colunas de caixas empilhadas. Cada caixa possui um peso inteiro positivo associado. O inspetor da fábrica precisa retirar algumas caixas da matriz de modo a deixar uma espécie de pirâmide de caixas satisfazendo as seguintes restrições:

- Se uma caixa está na pirâmide, a caixa imediatamente abaixo dela também deve estar na pirâmide;
- Na i-ésima linha de caixas (a linha 1 é a do topo da matriz), a pirâmide deve ter exatamente i caixas consecutivas.

Dados os pesos de todas as caixas na matriz, seu programa deve calcular o peso total mínimo que uma pirâmide poderá ter, se o inspetor retirar algumas caixas segundo as restrições acima.

Entrada

A primeira linha da entrada contém um inteiro N, indicando a dimensão da matriz. As N linhas seguintes contêm, cada uma, N inteiros representando os pesos das caixas em cada linha da matriz de caixas.

Saída

Seu programa deve produzir uma única linha, contendo um inteiro, indicando o peso total mínimo que a pirâmide poderá ter.

Restrições

- $1 \leq N \leq 100$
- Os valor dos elementos da matriz está entre 1 e 100, inclusive.

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 20 pontos, $N \leq 20$.

Exemplo de entrada 1	**Exemplo de saída 1**
3 5 2 4 3 6 7 10 5 10	36

Exemplo de entrada 2	**Exemplo de saída 2**
4 45 8 3 1 1 10 5 67 4 4 3 18 10 4 7 12	62