Olimpíada Brasileira de Informática

OBI2026

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 1

10 a 12 de Junho de 2026

A PROVA TEM DURAÇÃO DE 2 horas

Promoção:

Sociedade Brasileira de Computação

Apoio:

Coordenação:

v1.0

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

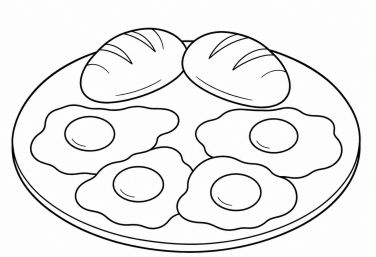
- Este caderno de tarefas é composto por 7 páginas (não contando a folha de rosto), numeradas de 1 a 7. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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.

Receita Revolucionária

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

Ao longo de sua vida, Murilo aprimorou suas habilidades culinárias e, enfim, descobriu uma receita revolucionária de sabor insuperável: pão com ovo. Esta combinação é sua maior obra-prima gastronômica, devido à versatilidade desses ingredientes. Os ovos podem ser preparados de diversas maneiras, como ovos fritos, cozidos, mexidos, pochê ou omeletes. Estes podem ser combinados com inúmeras variedades de pão, para criar diferentes pratos de café da manhã.

Existem infinitas possibilidades culinárias com esses ingredientes, e Murilo adora provar combinações de variedades de pão e ovo. Dessa forma, todo dia, no café da manhã, ele come **exatamente** 2 pães e 4 ovos.



exemplo de café da manhã de Murilo

No entanto, vendo que seu estoque havia acabado, o garoto foi ao supermercado SBC (Sobremesas, Bebidas e Comidas) e comprou P pães e O ovos.

Murilo está muito ansioso para testar novas receitas, mas se pergunta quantos dias esses ingredientes irão durar. Assim, dadas as quantidades P e O, ajude Murilo a calcular quantos cafés da manhã ele conseguirá tomar com os ingredientes que ele acabou de comprar.

Entrada

A primeira linha da entrada contém um único inteiro P, a quantidade de pães que Murilo comprou. A segunda linha da entrada contém um único inteiro O, a quantidade de ovos que Murilo comprou.

Saída

A saída deve conter um único inteiro, a quantidade de cafés da manhã que Murilo conseguirá tomar.

Restrições

- $1 \leq P \leq 20$.
- $1 \leq O \leq 30$.

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos)**: Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (100 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
10 13	3

Explicação do exemplo 1: Perceba que, para tomar café da manhã por 3 dias, Murilo precisa de 6 pães e 12 ovos, como ele comprou 10 pães e 13 ovos, essas quantidades são suficientes. Note também que, apesar de ter pães suficientes para 4 dias, Murilo não possui ovos para 4 dias.

Exemplo de entrada 2	**Exemplo de saída 2**
3 9	1

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

Explicação do exemplo 3: Note que Murilo não consegue tomar nenhum café da manhã com a quantidade de pães e ovos que ele comprou.

Elevador

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

Devido ao crescente destaque do país nas competições internacionais de programação, a OBI (Olimpíada Brasileira de Informática) passou por um processo de expansão e agora é dona de um prédio de 100 andares! No entanto, a construção é um pouco antiga e possui um erro básico em seu projeto: há apenas um elevador. Consequentemente, seus funcionários perdem muito tempo para se mover entre os andares do prédio, o que afeta a produtividade da empresa.

A fim de analisar o impacto desse problema, a OBI decidiu estudar o tempo gasto na movimentação do elevador ao longo de um dia de trabalho. Nesse período, o elevador fez N paradas em sequência, denotadas pela lista $A_1, A_2, \ldots, A_N$. O deslocamento entre essas paradas é simples e preciso, de forma que o tempo gasto para subir ou descer um andar é de **exatamente** 1 segundo. Além disso, para os propósitos dessa pesquisa, o tempo de entrada e saída de pessoas no elevador é irrelevante.

Por exemplo, para $N = 3$ e as seguintes paradas: $A_1 = 1, A_2 = 9$ e $A_3 = 5$, o elevador demoraria 8 segundos para subir do 1º ao 9º andar e, em seguida, 4 segundos para descer do 9º ao 5º andar. Ou seja, seu tempo total de deslocamento seria $8 + 4 = 12$ segundos.

Essa análise é muito importante para a OBI, mas seu departamento de pesquisa e desenvolvimento está muito ocupado com a criação de um novo ambiente de provas para a olimpíada. Assim, dada a quantidade N e a lista $A_1, A_2, \ldots, A_N$ de paradas do elevador, ajude a OBI a calcular por quanto tempo o elevador se moveu durante o dia.

Entrada

A primeira linha da entrada contém um único inteiro N, a quantidade de paradas.

A segunda linha da entrada contém N inteiros $A_1, A_2, \ldots, A_N$, os andares pelos quais o elevador passou ao longo do dia, em ordem.

Saída

A saída deve conter um único inteiro, o tempo total de deslocamento do elevador.

Restrições

- $3 \leq N \leq 100$.
- $1 \leq A_i \leq 100$, para todo $1 \leq i \leq N$.
- $A_i \neq A_{i+1}$, para todo $1 \leq i < N$.
- $A_1 = 1$, ou seja, o elevador sempre começa no andar 1.

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos)**: Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.

- **Subtarefa 2 (20 pontos):** $N = 3$.
- **Subtarefa 3 (30 pontos):** $A_i < A_{i+1}$, para todo $1 \leq i < N$.
- **Subtarefa 4 (50 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 1 9 5	12

Explicação do exemplo 1: Note que este é o exemplo mostrado no enunciado.

Exemplo de entrada 2	**Exemplo de saída 2**
4 1 7 9 6	11

Exemplo de entrada 3	**Exemplo de saída 3**
5 1 3 8 11 20	19

Explicação do exemplo 3: Perceba que este exemplo satisfaz as restrições da subtarefa 3.

Restaurante

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

Você trabalha no restaurante que faz o melhor pão com ovo da cidade, e por isso está sempre lotado. Sendo assim, você ficou responsável por cuidar da alocação das mesas do restaurante.

As mesas desse restaurante são sempre de 4 lugares, por conta disso, o sistema de reserva só aceita grupos de 1 a 4 pessoas. Devido a fama do restaurante, todos aceitam dividir mesas com pessoas desconhecidas, desde que as pessoas em um grupo se sentem na mesma mesa.

Sua tarefa é, dadas as quantidades de grupos com uma, duas, três e quatro pessoas, do sistema de reserva para o dia atual, determine a menor quantidade de mesas para alocar todos os grupos.

Entrada

A primeira e única linha da entrada contém quatro inteiros G_1, G_2, G_3, G_4, as quantidades de grupos com uma, duas, três, e quatro pessoas, respectivamente.

Saída

A saída deve conter uma única linha contendo um inteiro, representando a quantidade mínima de mesas necessária para alocar todos os grupos.

Restrições

É garantido que todo caso de teste satisfaz as restrições a seguir.

- $0 \leq G_1,\ G_2,\ G_3,\ G_4 \leq 10$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos)**: Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (20 pontos):** $G_2 = 0,\ G_3 = 0,\ G_4 = 0$, ou seja, só existem grupos de uma pessoa.
- **Subtarefa 3 (80 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`2 2 1 0`	`3`

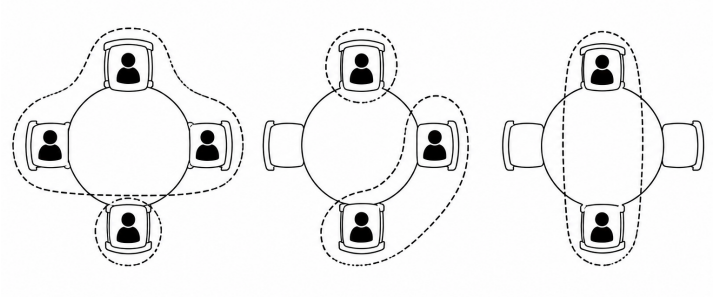
Explicação do exemplo 1: Nesse exemplo, temos:

- 2 grupos com uma pessoa
- 2 grupos com duas pessoas
- 1 grupo com três pessoas
- nenhum grupo com quatro pessoas

Com 3 mesas podemos alocar os grupos da seguinte forma:

- Mesa 1: um grupo com três pessoas e um grupo com uma pessoa
- Mesa 2: um grupo com uma pessoa e um grupo com duas pessoas
- Mesa 3: um grupo com duas pessoas

Conforme a imagem a seguir:



Não é possível alocar todos os grupos com menos de 3 mesas.

Exemplo de entrada 2	**Exemplo de saída 2**
7 0 0 0	2

Explicação do exemplo 2: Nesse exemplo temos apenas 7 grupos com uma pessoa cada.
Com 2 mesas podemos alocar os grupos da seguinte forma:

- Mesa 1: quatro grupos com uma pessoa
- Mesa 2: três grupos com uma pessoa

Perceba que esse exemplo satisfaz as restrições da subtarefa 2.

Exemplo de entrada 3	**Exemplo de saída 3**
0 0 3 4	7