Olimpíada Brasileira de Informática

OBI2026

Caderno de Tarefas

Modalidade Programação • Nível 1 • 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

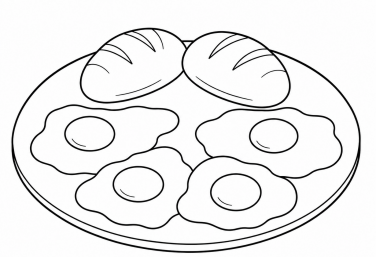
- Este caderno de tarefas é composto por 7 páginas (não contando a folha de rosto), numeradas de 1 a 7. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta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

Encontro de Amigas

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

Ana, Beatriz e Carolina são grandes amigas de infância e estudaram juntas a vida toda. Elas tinham notas excelentes e, além disso, competiram diversas vezes na Olimpíada Brasileira de Informática, ganhando diversas medalhas tanto nacionais quanto internacionais. Por causa disso, ao se formarem no ensino médio, as amigas foram aceitas em universidades ao redor de todo o mundo!

Elas comemoraram muito as aprovações, mas ficaram com medo de perderem o contato com o tempo. Isso porque, por estarem em universidades e países diferentes, a distância faz com que elas nem sempre possam se ver. Assim, Ana sugeriu que elas se encontrassem durante as férias de julho, quando estariam de volta no Brasil, para poderem colocar a conversa em dia. Beatriz e Carolina gostaram da ideia, mas repararam que, como cada amiga estará no país em um período de tempo diferente, marcar um encontro com **todas** presentes pode ser complicado.

Solucionar isso não seria difícil para programadoras experientes como elas, mas as garotas decidiram desafiar os atuais competidores da OBI a resolver esse problema. Assim, dados os intervalos de dias em que Ana, Beatriz e Carolina estarão no Brasil, calcule em quantos dias elas podem marcar um encontro com as três presentes.

Entrada

A primeira linha da entrada contém um inteiro A_1, o primeiro dia em que Ana estará no Brasil. A segunda linha da entrada contém um inteiro A_2, o último dia em que Ana estará no Brasil.

A terceira linha da entrada contém um inteiro B_1, o primeiro dia em que Beatriz estará no Brasil. A quarta linha da entrada contém um inteiro B_2, o último dia em que Beatriz estará no Brasil.

A quinta linha da entrada contém um inteiro C_1, o primeiro dia em que Carolina estará no Brasil. A sexta linha da entrada contém um inteiro C_2, o último dia em que Carolina estará no Brasil.

Saída

A saída deve conter uma única linha com um único inteiro, a quantidade de dias em que as amigas podem marcar um encontro com todas presentes.

Restrições

É garantido que todo caso de teste satisfaz as restrições abaixo.

- $1 \leq A_1 \leq A_2 \leq 31$
- $1 \leq B_1 \leq B_2 \leq 31$
- $1 \leq C_1 \leq C_2 \leq 31$

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 (30 pontos):** $A_1 = A_2$.
- **Subtarefa 3 (70 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
1 15 5 20 12 18	4

Explicação do exemplo 1: Nesse caso os únicos dias em que as três amigas podem estar juntas são: 12, 13, 14 e 15.

Exemplo de entrada 2	**Exemplo de saída 2**
7 7 1 5 10 30	0

Explicação do exemplo 2: Nesse caso, não há nenhum dia em que todas as amigas estarão no Brasil. Perceba que esse exemplo satisfaz as restrições da subtarefa 2.

Mercadinho

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

Você foi contratado como gerente do mercadinho do seu bairro, e sua primeira tarefa é criar a cultura de deixarem os idosos serem atendidos primeiro, pois, diferente dos mercados onde existem filas preferenciais, o mercadinho possui apenas um caixa com fila única. Vale ressaltar que idoso é toda pessoa com idade igual ou superior a 60 anos, conforme o Estatuto da Pessoa Idosa.

O objetivo é que todos os idosos estejam no início da fila, ou seja, todo idoso deve estar numa posição mais à frente na fila do que qualquer pessoa jovem. Para isso, os idosos podem ultrapassar a pessoa logo à sua frente. Realizar isso leva exatamente 1 segundo, e pode ser feito por vários idosos ao mesmo tempo. Por exemplo, se inicialmente as pessoas na fila tem 32, 75 e 67 anos, nesta ordem, em um único segundo os dois idosos podem se mover juntos, resultando na seguinte ordem de idades: 75, 67 e 32 anos.

Para demonstrar ao dono do mercadinho que o tempo para organizar a fila vale a pena, você deve escrever um programa que, dada a quantidade de pessoas na fila e a idade de cada pessoa, calcule o menor tempo, em segundos, para que todos os idosos estejam no início da fila.

Entrada

A primeira linha da entrada contém um único inteiro N, o número de pessoas na fila. A segunda linha da entrada contém N números inteiros $I_1, I_2, \ldots, I_N$, as idades das pessoas na fila, onde I_1 é a idade da primeira pessoa na fila, I_2 é a idade da segunda pessoa na fila, e assim por diante.

Saída

A saída deve conter um único inteiro, a quantidade mínima de segundos que levará para que todos os idosos estejam no início da fila.

Restrições

- $1 \leq N \leq 10^5$.
- $10 \leq I_i \leq 100$ para todo $1 \leq i \leq N$.

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 (40 pontos):** $N \leq 1000$.
- **Subtarefa 3 (20 pontos):** Existe um único idoso na fila.
- **Subtarefa 4 (40 pontos):** Nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
6 15 50 61 32 75 67	3

Explicação do exemplo 1: No primeiro segundo, os idosos nas posições 3, 5 e 6 ultrapassam as pessoas na sua frente. Assim, a idade das pessoas na fila, na ordem atual, é

$$[15, 61, 50, 75, 67, 32].$$

No próximo segundo,

$$[61, 15, 75, 67, 50, 32].$$

E finalmente, no terceiro segundo,

$$[61, 75, 67, 15, 50, 32].$$

Assim, demora 3 segundos para que todos os idosos fiquem no início da fila.

Exemplo de entrada 2	**Exemplo de saída 2**
7 10 11 13 12 67 10 10	4

Explicação do exemplo 2: Perceba que este caso satisfaz as restrições da subtarefa 3.

Exemplo de entrada 3	**Exemplo de saída 3**
5 43 23 22 11 59	0

Explicação do exemplo 3: Neste caso, não há idosos e portanto a resposta é 0.