



XIV Maratona de Programação da FATEC-Rubens Lara

1 de novembro de 2025

Caderno de Problemas

Este caderno contém 8 problemas; as páginas estão numeradas de 1 a 12. É proibido consultar a internet ou qualquer modelo de IA generativa. Apenas materiais impressos podem ser consultados.

Informações Gerais:

A) Sobre a entrada

1. A entrada de seu programa deve ser lida da entrada padrão.
2. A entrada é composta por um único caso de teste, descrito em um número de linhas que depende do problema.
3. Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco.
4. Cada linha, incluindo a última, contém o caractere final de linha.

B) Sobre a saída

1. A saída de seu programa deve ser escrita na saída padrão.
2. Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco; a saída não deve conter nenhum outro espaço em branco.
3. Cada linha, incluindo a última, deve conter o caractere final de linha.

Problema A

Computador Intcode

Nome do arquivo fonte: computador. [c | cpp | java | hs | py]

Durante uma tarde chuvosa na biblioteca da **Fatec**, você, um estudante curioso encontra, em meio a pilhas de manuais antigos, um misterioso livro de Arquitetura de Computadores. Entre diagramas de processadores e instruções em Assembly, há um trecho enigmático sobre uma antiga máquina chamada **Intcode**.

Movido pela curiosidade, o estudante decide reconstruir essa máquina em seu notebook, para entender como funcionavam os sistemas computacionais da era dos pioneiros. O aluno sem um rolê para o fim-de-semana, decidiu levar o livro para casa e reconstruir o **computador Intcode** do zero.

Descrição do problema:

Um **programa Intcode** é uma lista de números inteiros separados por vírgulas (por exemplo: 1,0,0,3,99). O programa deve ser executado passo a passo conforme as seguintes instruções:

- O primeiro número (posição 0) é o **opcode**.
- Os opcodes válidos são:
 - **1** — Soma dois valores;
 - **2** — Multiplica dois valores;
 - **99** — Finaliza o programa imediatamente.
- Após um opcode, seguem-se três números:
 - As duas primeiras posições indicam onde estão os valores de entrada;
 - A terceira indica onde o resultado deve ser armazenado.
- Depois de processar uma instrução (opcode + 3 parâmetros), o programa deve avançar 4 posições para ler a próxima.

Por exemplo, ao processar a sequência:

1, 9, 10, 3, 2, 3, 11, 0, 99, 30, 40, 50

A primeira instrução (1,9,10,3) soma o valor nas posições 9 e 10 ($30 + 40 = 70$) e armazena o resultado na posição 3. Depois, o programa avança 4 posições e encontra a instrução 2,3,11,0, que multiplica os valores nas posições 3 e 11 ($70 \times 50 = 3500$) e armazena o resultado na posição 0. Por fim, encontra o opcode 99 e encerra a execução. Caso o programa encontre um **opcode inválido** (diferente de 1, 2 ou 99) ou tente acessar uma posição que exceda o tamanho da memória (vetor), a execução deve ser considerada inválida, e o resultado final deve ser automaticamente **-1**. Assim como o estudante da Fatec, seu programa deve ser capaz de detectar esses erros e encerrar corretamente.

O programa deve ser executado até que o opcode 99 seja encontrado. Sua missão é determinar qual valor resta na posição 0 da memória após o término do programa.

Entrada:

Uma linha contendo a sequência de números inteiros $0 \leq I \leq 99$ separados por vírgulas, representando o programa Intcode.

Saída:

Um único número inteiro S , representando o valor armazenado na posição 0 após a execução completa do programa, ou -1 caso um erro seja encontrado.

Exemplo:

Entrada	Saída
1,0,0,0,99	2

Entrada	Saída
2,3,0,3,99	2

Entrada	Saída
2,4,4,5,99,0	2

Entrada	Saída
1,1,1,4,99,5,6,0,99	30

Entrada	Saída
1,9,10,3,2,3,11,0,99,30,40,50	3500

Entrada	Saída
1,5,0,0,99	-1

Problema B

Cônicas

Nome do arquivo fonte: *conicas*. [c | cpp | java | hs | py]

Autor: Prof. Dr. Lourval dos Santos Silva

Uma cônica é uma curva obtida pela interseção de um plano com um cone. Entre as mais conhecidas temos a **parábola**, a **elipse**, a **hipérbole** e a **circunferência**. Além dessas, também podem ocorrer casos degenerados, como o conjunto vazio, um ponto, duas retas concorrentes, uma reta ou duas retas paralelas.

Dada uma equação geral da forma:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

onde A, B, C, D, E, F são números reais (com A, B e C não sendo simultaneamente nulos), deseja-se identificar qual tipo de cônica está representada.

Para identificar a cônica, deve-se calcular os seguintes determinantes:

$$\begin{vmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{vmatrix} = D \quad \begin{vmatrix} A & B/2 \\ B/2 & C \end{vmatrix} = D_{33}$$

$$\begin{vmatrix} C & E/2 \\ E/2 & F \end{vmatrix} = D_{11},$$

$$\begin{vmatrix} A & D/2 \\ D/2 & F \end{vmatrix} = D_{22}$$

e o número

$$t = A + C$$

A partir dos valores de D_{ii} ($1 \leq i \leq 3$), D e t , é possível classificar a cônica conforme as condições da tabela a seguir:

Condição 1	Condição 2	Condição 3	Cônica
$D_{33} > 0$	$D \neq 0$	$t \cdot D > 0$	Conjunto vazio
$D_{33} > 0$	$D \neq 0$	$t \cdot D < 0, A = C, B = 0$	Circunferência
$D_{33} > 0$	$D \neq 0$	$t \cdot D < 0$	Elipse
$D_{33} < 0$	$D \neq 0$	—	Hipérbole
$D_{33} > 0$	$D = 0$	—	Um ponto
$D_{33} < 0$	$D = 0$	—	Duas retas concorrentes
$D_{33} = 0$	$D \neq 0$	—	Parábola
$D_{33} = 0$	$D = 0, D_{11} + D_{22} > 0$	—	Conjunto vazio
$D_{33} = 0$	$D = 0, D_{11} + D_{22} < 0$	—	Duas retas paralelas

Caso $A = B = C = 0$, não podemos ter uma cônica e este caso não será considerado.

Entrada:

Uma sequência contendo os valores reais A, B, C, D, E, F , nesta ordem, representando os coeficientes da equação geral da cônica.

Saída:

O nome da cônica correspondente, que pode ser: Conjunto vazio, Elipse, Circunferência, Hipérbole, Um ponto, Duas retas concorrentes, Parábola ou Duas retas paralelas.

Exemplos:

Entrada	Saída
1,0,1,0,0,-9	Circunferência

Entrada	Saída
1,0,-1,0,0,-16	Hipérbole

Entrada	Saída
1,0,0,0,-4,0	Parábola

Entrada	Saída
1,0,1,0,0,9	Conjunto vazio

Problema C

Adivinhe a Estrutura!

Nome do arquivo fonte: *estrutura*. [*c* | *cpp* | *java* | *hs* | *py*]

Durante um campeonato de programação da **Fatec**, os competidores precisam descobrir qual estrutura de dados está sendo utilizada por um programa misterioso. O programa executa duas operações, mas não revela se está usando uma **pilha**, uma **fila** ou uma **fila de prioridade**.

As operações são dadas em sequência:

- **1 x** — insere o número x na estrutura.
- **2 x** — remove um elemento da estrutura e verifica se o número removido é igual a x .

Com base nessas operações, sua tarefa é determinar qual estrutura de dados o programa poderia estar usando.

Você deve simular simultaneamente as três estruturas — **pilha**, **fila** e **fila de prioridade** — e verificar quais delas poderiam produzir a mesma sequência de remoções indicada pelas operações.

O resultado esperado é:

- **pilha** — se apenas a pilha for compatível.
- **fila** — se apenas a fila for compatível.
- **fila de prioridade** — se apenas a fila de prioridade for compatível.
- **inconclusivo** — se mais de uma estrutura puder ser compatível.
- **nenhum** — se nenhuma estrutura puder gerar a sequência dada.

Entrada:

A entrada contém vários casos de teste. Cada caso de teste começa com um número inteiro n ($1 \leq n \leq 1000$), indicando o número de operações seguintes. As n linhas seguintes contêm duas informações: o tipo de operação (1 ou 2) e o número $0 \leq x \leq 99$.

Saída:

Para cada caso de teste, imprima uma única linha contendo uma das palavras: **pilha**, **fila**, **fila de prioridade**, **inconclusivo** ou **nenhum**.

Exemplos:

Entrada	Saída
6 1 1 1 2 1 3 2 3 2 2 2 1	inconclusivo

Entrada	Saída
4 1 3 1 2 2 2 2 3	pilha

Entrada	Saída
6 1 1 1 2 1 3 2 1 2 2 2 3	fila

Entrada	Saída
6 1 3 1 1 1 2 2 3 2 2 2 1	fila de prioridade

Entrada	Saída
3 1 1 1 2 2 2	inconclusivo

Entrada	Saída
2 1 1 2 2	nenhum

Problema D

Descendentes

Nome do arquivo fonte: descendentes. [c | cpp | java | hs | py]

Em uma tarde tranquila na **Fatec**, você está ajudando um professor a organizar um sistema de genealogia usado em um projeto interdisciplinar. A ideia é simples: representar relações familiares, onde cada pessoa pode ter filhos, netos e até bisnetos.

O professor pede sua ajuda para completar o sistema, escrevendo um programa que informe quantos descendentes (filhos, netos, bisnetos, etc.) uma pessoa possui. Os dados são fornecidos na forma de pares de números, indicando quem é pai de quem.

Entrada:

- Um número inteiro N ($1 \leq N \leq 100$), representando quantas relações foram cadastradas;
- N linhas, cada uma com dois inteiros $1 \leq p \leq 50$ e $1 \leq f \leq 50$, indicando que p é genitor de f ;
- Um número inteiro $1 \leq R \leq 50$, representando a pessoa cujos descendentes devem ser contados.

Saída:

Um único número inteiro D , indicando quantos descendentes diretos e indiretos a pessoa R possui.

Exemplo:

Entrada	Saída
4 1 2 2 3 2 4 3 5 2	3

Problema E

O Robô Explorador

Nome do arquivo fonte: *robo.* [*c* | *cpp* | *java* | *hs* | *py*]

Em um dos laboratórios da **Fatec**, um grupo de alunos está programando um robô autônomo capaz de explorar uma área dividida em células, formando uma grade bidimensional. Cada célula contém um certo **valor de recompensa**, que o robô coleta ao passar por ela.

No entanto, o robô possui uma limitação importante: ele tem uma quantidade finita de **combustível**. A cada movimento — seja para a direita ou para baixo — ele consome exatamente uma unidade de combustível.

Sua missão é ajudar os estudantes a programar o sistema de controle do robô, determinando o **máximo valor total de recompensas** que o robô pode coletar dentro de sua limitação de combustível.

O robô sempre inicia no canto superior esquerdo da grade (posição $(0,0)$), e pode mover-se apenas para **direita** ou **baixo**. O combustível é consumido a cada movimento (uma unidade), e o robô não pode sair da grade.

Entrada:

- Três números inteiros N , M e F ($1 \leq N, M \leq 50$ e $1 \leq F \leq 200$), representando respectivamente o número de linhas, o número de colunas e o total de combustível disponível;
- N linhas seguintes, cada uma contendo M inteiros v_{ij} ($0 \leq v_{ij} \leq 100$), representando o valor de recompensa presente na célula (i, j) .

Saída:

Um único número inteiro V , representando o valor máximo total de recompensas coletadas pelo robô considerando todas as restrições.

Exemplo:

Entrada	Saída
3 3 4 1 3 1 1 9 1 4 2 1	16

Entrada	Saída
4 4 5 1 2 3 4 2 10 2 1 3 1 5 1 2 2 2 2	22

Problema F

Consultório Oftalmológico

Nome do arquivo fonte: *consultorio*. [c | cpp | java | hs | py]

Em uma manhã chuvosa na **Fatec**, um estudante curioso resolveu fazer um exame de vista, pois estava com dificuldade para distinguir letras na tela do computador. Ao chegar no consultório, o oftalmologista — um senhor esguio de aparência calma, mas com fama de exímio *hacker* — pediu que ele fizesse um simples teste: contar quantas letras **A** havia em uma sequência apresentada no monitor.

O estudante, confiante, começou a contar... mas errou feio. O doutor deu uma risadinha e explicou: *"Nem todas as letras que parecem 'A' são realmente 'A'. Algumas são caracteres cirílicos — idênticos ao olho humano, mas completamente diferentes para o computador."*

Sua missão é ajudar o estudante a não passar vergonha de novo. Dada uma string contendo letras do alfabeto latino e alguns caracteres que se parecem com elas, determine quantas letras '**A**' (não latino) estão presentes na string.

Entrada:

Uma linha contendo uma sequência de caracteres, incluindo letras maiúsculas e minúsculas do alfabeto latino e, possivelmente, caracteres de outros alfabetos (como o cirílico).

Saída:

Um único número inteiro C , representando a quantidade de letras **A** não latinas na string.

Exemplo:

Entrada	Saída
AAAAA	3

Entrada	Saída
ABCAAaA	2

Entrada	Saída
AAA	3

Entrada	Saída
AAA	0

Problema G

Serpente Dourada

Nome do arquivo fonte: *serpente*. [c | cpp | java | hs | py]

Em uma tarde silenciosa na biblioteca da **Fatec**, um estudante curioso encontrou um antigo livro chinês da dinastia Ming. Entre símbolos e anotações misteriosas, o texto mencionava alguns anos especiais chamados de “*Anos da Serpente Dourada*”.

Após pesquisar, o estudante descobriu que:

- O **ano do Serpente** ocorre a cada 12 anos (exemplos: 2001, 2013, 2025, ...);
- Alguns anos são considerados **dourados** (exemplos: 2004, 2011, 2018, 2025, ...). O padrão sempre se repete.

Dada uma lista de anos, determine quantos deles são ao mesmo tempo **da Serpente** e **dourado**.

Entrada:

- Um número inteiro N ($1 \leq N \leq 100$), representando quantos anos foram listados;
- N inteiros Y_i ($0 \leq Y_i \leq 9999$), representando os anos.

Saída:

- Um único número inteiro K , indicando quantos dos anos fornecidos são **Anos do Serpente Dourada**.

Exemplos:

Entrada	Saída
5 930 1857 1941 2025 2026	3

Entrada	Saída
4 287 1100 1773 1800	1

Problema H

Notas do Professor

Nome do arquivo fonte: *notas*. [*c* | *cpp* | *java* | *hs* | *py*]

Em uma tarde movimentada na **Fatec**, o professor de Estruturas de Dados está corrigindo pilhas de listas de exercícios. Ele possui K **monitores** (assistentes de ensino), e cada um deles entregou uma lista com as notas dos alunos, **já ordenadas do menor para o maior**.

O professor, porém, quer divulgar o ranking final de desempenho — do **maior para o menor**. Como as listas são muito grandes, ele não quer juntar tudo manualmente: precisa de um algoritmo eficiente para fundir as K listas e produzir a sequência final.

Escreva um programa que receba as listas de notas entregues pelos monitores e imprima todas as notas em ordem decrescente, como o professor deseja.

Entrada:

- Um inteiro K ($1 \leq K \leq 50$), o número de monitores;
- Para cada monitor i ($1 \leq i \leq K$):
 - Um inteiro N_i ($1 \leq N_i \leq 10^3$), o número de notas desse monitor;
 - Uma linha com N_i inteiros $a_{i,1}, a_{i,2}, \dots, a_{i,N_i}$ ($0 \leq a_{i,j} \leq 100$), em ordem crescente.

Saída:

- Todos os valores $a_{i,j}$ em uma única linha, separados por espaços, em **ordem decrescente**.

Exemplo:

Entrada	Saída
3 4 1 3 5 7 3 2 6 10 5 0 2 4 8 9	10 9 8 7 6 5 4 3 2 2 1 0