Centro de Educação Superior a Distância do
Estado do Rio de Janeiro – CEDERJ

Curso de Tecnologia em Sistemas de Computação – TSC

EAD-05.009 Fundamentos de Programação

Caderno de Exercícios
Aula 12

(Persistência de Dados: Arquivos Binários)

Professores

Dante Corbucci Filho
Leandro A. F. Fernandes

Instruções

- Utilize Python 3 e a IDE PyCharm na elaboração de soluções para os problemas propostos;
- A entrada de cada problema deve ser lida da entrada padrão (teclado);
- A saída de cada problema deve ser escrita na saída padrão (tela);
- Siga o formato apresentado na descrição da saída, caso contrário não é garantido que a saída emitida será considerada correta;
- Na saída, toda linha deve terminar com o caractere `'\n';`
- Utilize o URI Online Judge (http://www.urionlinejudge.com.br) e submeta sua solução para correção automática.

Referências Autorais

Os exercícios apresentados nesta lista foram extraídos do URI Online Judge (http://www.urionlinejudge.com.br). Acesse a URL apresentada abaixo do título de cada problema para proceder com a correção automática de sua solução e, também, para consultar a autoria do enunciado.

Problema A: Aviões de Papel

https://www.urionlinejudge.com.br/judge/pt/problems/view/2339

Observação: Após o seu programa funcionar no URI, carregue vários valores de entrada através de um arquivo chamado entrada.bin e a sua saída deve ser escrita no arquivo saida.bin.

Para descontrair os alunos após as provas da OBI, a Diretora da escola organizou um campeonato de aviões de papel. Cada aluno participante receberá uma certa quantidade de folhas de um papel especial para fazer os seus modelos de aviões. A quantidade de folhas que cada aluno deverá receber ainda não foi determinada: ela será decidida pelos juízes do campeonato.

A diretora convidou, para atuarem como juízes, engenheiros da Embraer, uma das mais bem sucedidas empresas brasileiras, que vende aviões com tecnologia brasileira no mundo todo. O campeonato está programado para começar logo após a prova da OBI, mas os juízes ainda não chegaram à escola. A diretora está aflita, pois comprou uma boa quantidade de folhas de papel especial, mas não sabe se a quantidade comprada vai ser suficiente.

Considere, por exemplo, que a Diretora comprou 100 folhas de papel especial, e que há 33 competidores. Se os juízes decidirem que cada competidor tem direito a três folhas de papel, a quantidade comprada pela diretora é suficiente. Mas se os juízes decidirem que cada competidor tem direito a quatro folhas, a quantidade comprada pela diretora não seria suficiente.

Você deve escrever um programa que, dados o número de competidores, o número de folhas de papel especial compradas pela Diretora e o número de folhas que cada competidor deve receber, determine se o número de folhas comprado pela Diretora é suficiente.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do dispositivo de entrada padrão (normalmente o teclado). O arquivo de entrada contém três números inteiros **C** ($1 \leq \mathbf{C} \leq 1000$), **P** ($1 \leq \mathbf{P} \leq 1000$) e **F** ($1 \leq \mathbf{F} \leq 1000$) representando respectivamente o número de competidores, a quantidade de folhas de papel especial compradas pela Diretora e a quantidade de folhas de papel especial que cada competidor deve receber.

Saída

Seu programa deve imprimir, na saída padrão, o caractere ‘S’ se a quantidade de folhas compradas pela Diretora é suficiente, ou o caractere ‘N’ caso contrário. Note que os caracteres devem ser letras maiúsculas.

Exemplo

Entrada	Saída
10 100 10	S

Entrada	Saída
10 90 10	N

Entrada	Saída
5 40 2	S

Problema B: Lanche

https://www.urionlinejudge.com.br/judge/pt/problems/view/2295

Observação: Após o seu programa funcionar no URI, carregue vários valores de entrada através de um arquivo chamado entrada.bin e a sua saída deve ser escrita no arquivo saida.bin.

A Companhia de Táxi Tabajara (CTT) é uma das maiores empresas de transporte do país. Possui uma vasta frota de carros e opera em todas as grandes cidades. Recentemente a CTT modernizou a sua frota, adquirindo um lote de 500 carros bi-combustíveis (carros que podem utilizar como combustível tanto álcool quanto gasolina). Além do maior conforto para os passageiros e o menor gasto com manutenção, com os novos carros é possível uma redução adicional de custo: como o preço da gasolina está sujeito a variações muito bruscas e pode ser vantagem, em certos momentos, utilizar álcool como combustível. Entretanto, os carros possuem um melhor desempenho utilizando gasolina, ou seja, em geral, um carro percorre mais quilômetros por litro de gasolina do que por litro de álcool.

Você deve escrever um programa que, dados o preço do litro de álcool, o preço do litro de gasolina e os quilômetros por litro que um carro bi-combustível realiza com cada um desses combustíveis, determine se é mais econômico abastecer os carros da CTT com álcool ou com gasolina. No caso de não haver diferença de custo entre abastecer com álcool ou gasolina a CTT prefere utilizar gasolina.

Entrada

A entrada é composta por uma linha contendo quatro números reais com precisão de duas casas decimais **A** e **G** ($0.01 \leq \mathbf{A}, \mathbf{G} \leq 10.00$) $\mathbf{R_a}$ e $\mathbf{R_g}$ ($0.01 \leq \mathbf{R_a}, \mathbf{R_g} \leq 20.00$) representando respectivamente o preço por litro do álcool, o preço por litro da gasolina, o rendimento (km/l) do carro utilizando álcool e o rendimento (km/l) do carro utilizando gasolina.

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

A saída deve ser composta por uma única linha contendo o caractere ‘A’ se é mais econômico abastecer a frota com álcool ou o caractere ‘G’ se é mais econômico ou indiferente abastecer a frota com gasolina.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Exemplo

Entrada	Saída
1.20 2.30 10.00 15.00	A

Entrada	Saída
1.00 1.00 9.00 9.01	G

Entrada	Saída
1.00 1.00 11.00 11.00	G

Problema C: Sort Simples

https://www.urionlinejudge.com.br/judge/pt/problems/view/2297

Observação: Após o seu programa funcionar no URI, carregue vários valores de entrada através de um arquivo chamado entrada.bin e a sua saída deve ser escrita no arquivo saida.bin.

Álbuns de figurinhas – sejam de times de futebol, princesas ou super-heróis – têm marcado gerações de crianças e adolescentes. Conseguir completar um álbum é uma tarefa muitas vezes árdua, envolvendo negociações com colegas para a troca de figurinhas. Mas a existência das figurinhas propicia uma outra brincadeira, que foi muito popular entre crianças no século passado: o jogo de bater figurinhas (o famoso "Bafo"). O jogo é muito simples, mas divertido (e muito competitivo). No início de uma partida, cada criança coloca em uma pilha um certo número de figurinhas. Uma partida é composta de rodadas; a cada rodada as crianças batem com a mão sobre a pilha de figurinhas, tentando virá-las com o vácuo formado pelo movimento da mão. As crianças jogam em turnos, até que a pilha de figurinhas esteja vazia. Ganha a partida a criança que conseguir virar mais figurinhas.

Aldo e Beto estão jogando bafo com todas as suas figurinhas e pediram sua ajuda para calcular quem é o vencedor.

Você deve escrever um programa que, dada a quantidade de figurinhas que Aldo e Beto viraram em cada rodada, determine qual dos dois é o vencedor.

Entrada

A entrada é composta de vários casos de teste, cada um correspondendo a uma partida entre Aldo e Beto. A primeira linha de um caso de teste contém um número inteiro **R** ($1 \leq \mathbf{R} \leq 1000$) que indica quantas rodadas ocorreram na partida. Cada uma das **R** linhas seguintes contém dois inteiros, **A** e **B** ($0 \leq \mathbf{A}, \mathbf{B} \leq 100$), que correspondem, respectivamente, ao número de figurinhas que Aldo e Beto conseguiram virar naquela rodada. Em todos os casos de teste há um único vencedor (ou seja, não ocorre empate). O final da entrada é indicado por $\mathbf{R} = 0$.

Saída

Para cada caso de teste da entrada, seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do caso de teste, no formato "Teste n", onde n é numerado sequencialmente a partir de 1. A segunda linha deve conter o nome do vencedor (Aldo ou Beto). A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo

Entrada	Saída
2 1 5 2 3 3 0 0 4 7 10 0 0	Teste 1 Beto Teste 2 Aldo

Problema D: Pares entre Cinco Números

https://www.urionlinejudge.com.br/judge/pt/problems/view/2321

Observação: Após o seu programa funcionar no URI, carregue vários valores de entrada através de um arquivo chamado entrada.bin e a sua saída deve ser escrita no arquivo saída.bin.

Detecção de colisão é uma das operações mais comuns (e importantes) em jogos eletrônicos. O objetivo, basicamente, é verificar se dois objetos quaisquer colidiram, ou seja, se a interseção entre eles é diferente de vazio. Isso pode ser usado para saber se duas naves colidiram, se um monstro bateu numa parede, se um personagem pegou um item, etc.

Para facilitar as coisas, muitas vezes os objetos são aproximados por figuras geométricas simples (esferas, paralelepípedos, triângulos etc). Neste problema, os objetos são aproximados por retângulos num plano 2D.

Escreva um programa que, dados dois retângulos, determine se eles se interceptam ou não.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). Cada caso de teste contém duas linhas. Cada linha contém quatro inteiros (x_0, y_0, x_1, y_1, sendo $0 \leq x_0 < x_1 \leq 1.000.000$ e $0 \leq y_0 < y_1 \leq 1.000.000$) separados por um espaço em branco representando um retângulo. Os lados do retângulo são sempre paralelos aos eixos *x* e *y*.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha para cada caso de teste, contendo o número 0 (zero) caso não haja interseção ou o número 1 (um) caso haja.

Exemplo

Entrada	Saída
0 0 1 1 0 0 1 1	1

Entrada	Saída
0 0 2 2 1 1 3 3	1

Entrada	Saída
0 0 1 1 2 2 3 3	0

Problema E: Peça Perdida

https://www.urionlinejudge.com.br/judge/pt/problems/view/2322

Observação: Após o seu programa funcionar no URI, carregue vários valores de entrada através de um arquivo chamado entrada.bin e a sua saída deve ser escrita no arquivo saída.bin.

Joãozinho adora quebra-cabeças, essa é sua brincadeira favorita. O grande problema, porém, é que às vezes o jogo vem com uma peça faltando. Isso irrita bastante o pobre menino, que tem de descobrir qual peça está faltando e solicitar uma peça de reposição ao fabricante do jogo. Sabendo que o quebra-cabeças tem *N* peças, numeradas de 1 a *N* e que exatamente uma está faltando, ajude Joãozinho a saber qual peça ele tem de pedir.

Escreva um programa que, dado um inteiro *N* e *N* - 1 inteiros numerados de 1 a *N*, descubra qual inteiro está faltando.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A entrada contém 2 linhas. A primeira linha contém um inteiro N ($2 \leq N \leq 1.000$). A segunda linha contém N - 1 inteiros numerados de 1 a N (sem repetições).

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo o número que está faltando na sequência dada.

Exemplo

Entrada	Saída
3 3 1	2

Entrada	Saída
5 1 2 3 5	4

Entrada	Saída
4 2 4 3	1