INSTITUTO FEDERAL
Paraíba
Campus Guarabira

LACED
Laboratório de Computação Embarcada e Distribuída

ORI
Olimpíada Regional de Informática

MODALIDADE PROGRAMAÇÃO
2ª FASE

A PROVA TERÁ DURAÇÃO DE **DUAS HORAS E TRINTA MINUTOS**

Este Caderno contém 6 problemas
24 de Maio de 2019

Instruções

LEIA ATENTAMENTE AS INSTRUÇÕES ABAIXO ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto de 10 páginas . 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 menu ajuda 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.
- 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 ordenadas, neste caderno, por ordem de dificuldade; procure resolver as mais fáceis primeiro.
- 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 .cpp; soluções na linguagem Java devem ser arquivos com sufixo . java e a classe principal deve ter o mesmo nome do arquivo fonte; e soluções na linguagem Python 2.7 devem ser arquivos com sufixo .py.
- 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 entradas e saída de dados:
    - em C: scanf, getchar, printf, putchar;
    - em Pascal: readln, read, writeln, write;
    - em C++: as mesmas do 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 2.7 ou Python 3.6 : read, readline, readlines, print, write, raw_input e input.**

- Procure resolver o problema de maneira eficiente. Na correção, Eficiência também será levada em conta. Em cada problema é indicado o tempo limite. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.
- Para a realização da prova entre no endereço <https://boca.laced.com.br/>, faça seu login e boa sorte!

Problema A. Criptografia da Tia Juice

Nome do Programa: A.(c | pas | cpp | java | py)
Tempo: 1 segundo

O computador da tia Juice é muito usado por ela e seus colegas. Para assegurar que seus documentos importantes não serão lidos por alguns invasores, tia Juice decidiu criptografar o nome dos arquivos, como forma de segurança para tais. Ela utiliza um método estranho de criptografia, que consiste em:

- Todas as letras do nome do documento tem que ser transformadas em um número correspondente do alfabeto, como a=1, b=2, d=4, etc.
- Depois de transformar a letra em número, deve-se separar da próxima com um "_".
- A extensão do arquivo deve ser ".juice".

Ajude-a nessa missão de segurança. Agora é com você.

Entrada

A primeira linha contém um inteiro positivo N(1 ≤ **N** ≤ 100), que corresponde ao número de arquivos no computador da tia Juice. As N linhas seguintes serão de uma string S, formada apenas por letras minúsculas, contendo o nome do arquivo.

Saída

Seu programa deve imprimir o nome do arquivo da tia Juice após a criptografia.

Exemplos de Entrada

Entrada	Saída
1 ori	15_18_9.juice

Entrada	Saída
3 ouro jarro ariel	15_21_18_15.juice 10_1_18_18_15.juice 1_18_9_5_12.juice

Problema B. Crimpagem do pai Erick

Nome do Programa: B.(c | pas | cpp | java | py)
Tempo: 1 segundo

Pai Erick era um grande mestre quando se tratava de crimpagem de cabos. Para transmitir os dados referentes à rede no Laboratório da Olimpíada Regional de Informática(LaORI), ele ensinou aos seus alunos do último período do curso de Informática a crimpar os cabos de rede. O processo consiste em organizar as pontas usando a sequência de cores contidas, seguindo algumas condições:

- As cores podem ser: Verde, Azul, Marrom e Laranja, sendo correspondidas como V, A, M, L, respectivamente, e seus correspondentes cor-brancos são, respectivamente, VB, AB, MB, LB.
- As duas pontas tem que ter a mesma sequência de cores.
- Para cada cor, como verde, tem sua correspondente, verde-branco, que, nesse padrão, devem ser colocado ao lado, com outra cor e outro cor-branco e assim sucessivamente.

Seus alunos sempre perguntam ao pai Erick se suas sequências estão corretas e você foi escolhido para ajudá-lo, verificando assim se as pontas dos cabos estão ordenadas seguindo as regras e se podem ser crimpadas.

Entrada

A entrada consiste em 2 duas linhas, cada uma com uma sequência de 8 strings separadas por um espaço, contendo as cores organizadas pelo aluno, das pontas de cada lado do cabo.

Saída

Seu programa deve imprimir “Sim” caso a sequência siga os padrões e “Nao” caso contrário.

Exemplos de Entrada

Entrada	Saída
V VB M MB L LB A AB V VB M MB L LB A AB	Sim

Entrada	Saída
V VB M MB A LB A AB V VB M MB L LB A AB	Nao

Problema C. Prendedores de roupas

Nome do Programa: C.(c | pas | cpp | java | py)
Tempo: 2 segundo

Era um belo dia ensolarado quando Railson decidiu lavar suas roupas. No momento em que começou a estendê-las ao sol para secar, percebeu que a quantidade de prendedores de roupas poderia ser insuficiente para pendurar todas as peças no varal. Então, ele percebeu que cada peça necessitava de uma quantidade diferente de prendedores, sendo compreendido como:

- Cada par de meias: 1 prendedor
- Cada calça: 2 prendedores
- Cada blusa: 2 prendedores, onde, para um conjunto com 2 blusas seriam necessários 3 prendedores, e assim por diante.

Dado o número de prendedores que ele possui, a quantidade de pares de meias, calças e de blusas que Railson deseja colocar para secar, determine se é possível estender todas as roupas no varal ou não.

Entrada

A primeira linha contém um inteiro positivo **P**(1 ≤ **P** ≤ 10.000), que representa o número de prendedores que Railson possui.
A próxima linha contém três inteiros positivos **M**(1 ≤ **M** ≤ 500), **C**(1 ≤ **C** ≤ 500), **B**(1 ≤ **B** ≤ 500), separados por um espaço, representando, respectivamente, a quantidade de pares de meias, calças e blusas que precisam ser estendidas, sendo obrigatório ao menos uma peça de blusa.

Saída

Seu programa deve imprimir uma linha contendo “Y” caso for possível e “N” caso contrário.

Exemplos de Entrada

Entrada	Saída
44 3 2 3	Y

Entrada	Saída
66 10 14 30	N

Problema D. Gabriel e o XOR perfeito

Nome do Programa: D.(c | pas | cpp | java | py)
Tempo: 1 segundo

Gabriel é uma pessoa que adora a operação bitwise XOR (OU EXCLUSIVO) e também adora matrizes. Como ele passava muito tempo livre entre as aulas de cálculo e de Algoritmos Avançados decidiu criar um jogo, ele escolhe dois valores N e M e então começa a preencher uma matriz de N linhas e M colunas com valores aleatórios, logo depois, ele escolhe exatamente 1 elemento de cada linha da matriz e dá o XOR de todos esses elementos escolhidos. Caso o XOR de todos elementos da matriz resulte em um XOR diferente de 0, Gabriel se considera vitorioso.

Uma amiga de Gabriel, Vitória, acha que é muito simples ganhar esse jogo e resolveu aumentar o tamanho da matriz e também passou a escolher os valores que iriam preencher a matriz, como é uma matriz muito grande e os valores estão sendo escolhidas por Vitória, ele suspeita que nem sempre é possível vencer.

Como ele não sabe muito sobre computadores e programação, pediu sua ajuda para que você fizesse um programa que dada a matriz do jogo, cheque se existe algum movimento que torna a vitória possível.

Entrada

Dois inteiros N e M (1 <= N, M <= 500) que determinam a dimensão da matriz.

As próximas N linhas contém M inteiros que são os valores da matriz, os valores dentro da matriz são <= 1023.

Saída

Escreva “SIM” se é possível vencer naquela matriz e “NAO” caso contrário.

Exemplos de Entrada

Entrada	Saída
3 2 0 0 0 0 0 0	NAO

Entrada	Saída
2 3 7 7 7 7 7 10	SIM

Dica: XOR é uma operação binária que resulta em 1 quando os 2 bits são diferentes e em 0 quando os bits são iguais, ela em forma de BITWISE é realizar essa operação binária na representação binária de 2 números. Na maioria das linguagens de programação para se fazer BITWISE XOR basta fazer C = A ^ B (C vai receber o valor Bitwise XOR de A e B).

Problema E. Extremamente !Básico

Nome do Programa: E.(c | pas | cpp | java | py)
Tempo: 2 segundo

Você recebe um sequência de N inteiros.

Determine um valor X tal que ao subtrair X de todos elementos da sequência, a *facilidade* é a menor possível.
A *facilidade* de uma sequência é definida pelo maior valor de *simplicidade* entre todos os possíveis segmentos contínuos da sequência.
A *simplicidade* de um segmento é definido como o valor absoluto da soma de todos os elementos do segmento.

Entrada

A primeira linha contém um inteiro 1 <= N <= 2*10^5 que é o tamanho da sequência.
A segunda linha contém N inteiros que podemos chamar de ni (-10^4 < ni < 10^4).

Saída

A saída deve ser a menor *facilidade* possível da sequência. Só é necessário imprimir a parte inteira da resposta.

Exemplos de Entrada

Entrada	Saída
3 1 2 3	1

Entrada	Saída
4 1 2 3 4	2

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

Obs: Note que a menor facilidade para o exemplo 3 seria 4.5, mas como só é necessário mostrar a parte inteira a resposta se torna 4.

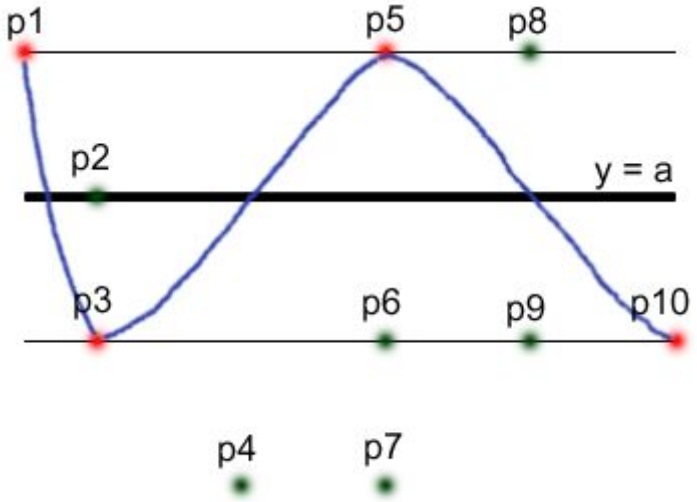
Problema F. Onda Crítica

Nome do Programa: F.(c | pas | cpp | java | py)
Tempo: 2 segundo

A tarefa é simples. Através de alguns pontos críticos em 2D, você deve desenhar uma onda como uma curva. Seu objetivo é incluir tantos pontos quantos forem possível.

- Haverá uma linha imaginária **y = a**, a qual chamaremos de eixo principal para a curva.
- Todos os pontos na curva deverão ter diferentes coordenadas x. Suas coordenadas y devem ser na forma **a-1** ou **a+1**.

Dois pontos consecutivos na curva devem ter **diferença de 2 na coordenada y**.



Entrada

Haverá no máximo, 222 casos de testes. Cada caso inicia com um inteiro N, que é o número de pontos no caso de teste. Nas próximas N linhas, haverá N pares de inteiros dando as coordenadas x e y de cada ponto. Não haverá mais do que 1000 pontos em cada caso de teste. Todas coordenadas são inteiros – eles devem ficar dentro de um inteiro de 2 bytes com sinal. Os dados devem ser lidos da entrada padrão.

Saída

Para cada caso de teste, imprima um número – o número máximo de pontos críticos que podem ser incluídos em uma curva desenhada a partir dos pontos dados.

Exemplos de Entrada

Entrada	Saída
10 0 1 1 0 1 -1 2 -2 3 1 3 -1 3 -2 4 1 4 -1 5 -1 10 0 2 2 0 1 -1 2 -2 3 1 3 -1 3 -2 4 1 4 -1 5 –1	4 3