13a. EDIÇÃO

MARATONA DE PROGRAMAÇÃO

InterFatecs

1a. Fase

18 de maio de 2024

CADERNO DE QUESTÕES

ORGANIZAÇÃO E APOIO

Fatec Bragança Paulista
Jornalista Omair Fagundes de Oliveira

Centro Paula Souza

SÃO PAULO
GOVERNO DO ESTADO
SÃO PAULO SÃO TODOS

WWW.INTERFATECS.COM.BR

1 Instruções

Este caderno contém 10 problemas – identificados por letras de A até L, com páginas numeradas de 3 até 30. Verifique se seu caderno está completo.

Informações gerais

1. Sobre a competição
   (a) A competição possui duração de 5 horas (início as 13:00h término as 18:00h);
   (b) NÃO é permitido acesso a conteúdo da Internet ou qualquer outro meio eletrônico digital;
   (c) NÃO é permitido o uso de ferramentas de auxílio à codificação, como GitHub Copilot, Tabnine, Amazon CodeWhisperer ou similar;
   (d) É permitido somente acesso a conteúdo impresso em papel (cadernos, apostilas, livros);
   (e) Não é permitida a comunicação com o técnico ou qualquer outra pessoa que não seja a equipe para tirar dúvidas sobre a maratona
   (f) Cada equipe terá acesso a 1 computador dotado do ambiente de submissão de programas (BOCA), dos compiladores, link-editores e IDEs requeridos pelas linguagens de programação permitidas;
   (g) NÃO é permitido o uso de notebooks, smartphones, ou outro tipo de computador ou assistente pessoal;
   (h) Todos os problemas têm o mesmo valor na correção.
2. Sobre o arquivo de solução e submissão:
   (a) O arquivo de solução (o programa fonte) DEVE TER o mesmo nome que o especificado no enunciado (logo após o título do problema);
   (b) confirme se você escolheu a linguagem correta e está com o nome de arquivo correto antes de submeter a sua solução;
   (c) NÃO insira acentos ou outros caracteres especiais no arquivo-fonte.
3. Sobre a entrada
   (a) A entrada de seu programa deve ser lida da entrada padrão (não use interface gráfica);
   (b) Seu programa será testado em vários casos de teste válidos além daqueles apresentados nos exemplos. Considere que seu programa será executado uma vez para cada caso de teste.
4. Sobre a saída
   (a) A saída do seu programa deve ser escrita na saída padrão;
   (b) Não exiba qualquer outra mensagem além do especificado no enunciado.
5. Versões das linguagens
   (a) gcc version 11.3.0 (lembre-se que não existem bibliotecas do Windows)
   (b) Python 3.10.6
   (c) Java 17.0.7+7-Ubuntu-0ubuntu122.04.2 (lembre-se que as classes devem estar fora de pacotes)

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema A

Mapa de calor

Arquivo fonte: mapacalor.{ c | cpp | java | py }
Autor: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

Munarinho descobriu recentemente sua paixão por *design* e não para de pensar sobre como o diálogo estabelecido entre um observador e um objeto observado dizem muito sobre os dois. Antes um pouco desleixado com o aspecto visual das coisas e com a aparência, decidiu agora cuidar de cada detalhe. Recentemente, fez um corte de cabelo em homenagem ao Cascão, assim como fez Ronaldo Fenômeno na Copa de 2002.

Em uma de suas explorações sobre a área, Munarinho descobriu sobre a análise do comportamento de usuários em páginas web. Após estudar os artigos do Nielsen Norman Group, resolveu prontamente analisar o comportamento do olhar dos usuários na página de sua Fatec. Estava convencido que o padrão F-Shaped estava presente em muitas interações e receava que isso não fosse bom.

Um padrão de leitura de uma página web pode ser obtido a partir do rastreamento do olhar de vários usuários enquanto interagem com a página e da produção de um mapa de calor como o seguinte.

Figura A.1: Imagem de mapa de calor retirada de https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content

Este mapa de calor indica um padrão conhecido como F-Shaped, em que o olhar dos usuários forma, algumas vezes, um formato que lembra a letra F. Procurando realizar uma primeira avaliação do site de sua Fatec, Munarinho está desenvolvendo um sistema que procura identificar a área de uma página que os usuários mais olharam: superior, esquerda, centro, direita ou inferior.

Para simplificar o processamento, ele projetou os pontos que os usuários olharam em uma interação em uma matriz de 6x3. Nesta matriz, os pontos que um usuário observou são representados por 1 e os não observados por 0. Ainda, as seguintes regiões são de interesse de Munarinho:

- Região: (coluna inicial, linha inicial, coluna final, linha final)
- Superior: (0, 0, 2, 0)

- Esquerda: (0, 1, 0, 4)
- Centro: (1, 1, 1, 4)
- Direita: (2, 1, 2, 4)
- Inferior: (0, 5, 2, 5)

Entrada

A entrada é composta por um caso de teste. A primeira linha do caso de teste contém um inteiro $N(3 \leq N \leq 100)$, indicando o número de usuários que interagiram com a página. As próximas $N*6$ linhas contém 3 inteiros $P(0 \leq P \leq 1)$, separados por espaços em branco, cada indicando se o usuário olhou ou não para a posição em questão.

Saída

Imprima o nome da região, em minúsculas e sem caracteres especiais, que recebeu mais olhares dos usuários (cada olhar soma 1 na região). Caso haja empate, imprima a região que aparece antes nos itens apresentados no enunciado.

Exemplo de Entrada 1	Exemplo de Saída 1
3 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1	inferior

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema B

Teclado

Arquivo fonte: teclado.{ c | cpp | java | py }
Autor: Prof. Dr. Reinaldo Arakaki(Fatec São José dos Campos)

O teclado de um celular possui associado a ele números e letras, como mostra a figura seguinte. Joãozinho recebeu a seguinte tarefa: dadas algumas palavras, descobrir qual o número de telefone cada representa.

Entrada
A primeira linha contém um inteiro N, ($1 \leq N \leq 100$), indicando o número de palavras. As próximas N linhas contêm palavras em letras maiúsculas.

Saída
Imprima os telefones correspondentes a estas palavras.

Exemplos

Exemplo de Entrada 1	Exemplo de Saída 1
3 PORTOSEGURO TRICOLOR CENTROPAULASOUZA	76786734876 87426567 2368767285276892

Exemplo de Entrada 2	Exemplo de Saída 2
1 FATEC	32832

Exemplo de Entrada 3	Exemplo de Saída 3
2 SOS AMBULANCIA	767 2628526242

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema C

Calcetis

Arquivo fonte: calcetis.{ c | cpp | java | py }
Autor: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

Embalado por um movimento crescente de investimento em startups orientais, o neto de Munarinho, Tikomo Nakama, decidiu abrir uma startup, Calcetis, com a ideia de vender calcinhas usadas por musas do cinema, televisão e redes sociais.

Ele está em uma etapa importante do desenvolvimento em que precisa sugerir novos produtos aos compradores que já têm produtos no carrinho de compras. A ideia é que o valor do carrinho some R$ 200,00, para que o comprador tenha a opção de não pagar o frete. Por uma questão de simbologia oriental, que trará sorte para a startup, Nakama deseja ofertar exatamente 3 produtos diferentes que somem, juntos aos produtos do carrinho, exatamente o valor necessário, caso seja possível.

Ajude Nakama - e conte com a gratidão de Munarinho - escrevendo um programa de computador **eficiente** para isso.

Entrada

A entrada é composta por um caso de teste. A primeira linha do caso de teste contém dois inteiros $V(30 \leq V < 200)$ e $N(3 \leq N \leq 10^5)$, separados por um espaço em branco, indicando o valor atual do carrinho de compras e o número de produtos vendidos na Calcetis que ainda não estão no carrinho, respectivamente. As próximas N linhas contém um inteiro $P(30 \leq P \leq 200)$ cada indicando o preço de um dos N produtos que não estão no carrinho de compras.

Saída

Para o valor atual do carrinho e os preços de produtos apresentados, imprima $fretegratis$ caso seja possível encontrar 3 produtos diferentes que, quando tiverem uma (1) unidade de cada adicionada ao carrinho, somarão o valor necessário para que o comprador tenha a opção de não pagar o frete. Caso contrário, imprima $fretepago$.

Exemplo de Entrada 1	Exemplo de Saída 1
52 10 50 30 33 91 68 40 30 32 41 39	fretegratis

MARATONA DE PROGRAMAÇÃO
InterFatecs

Exemplo de Entrada 2	Exemplo de Saída 2
34 10 50 30 33 91 68 40 30 32 41 38	fretepago

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema D

Rotação

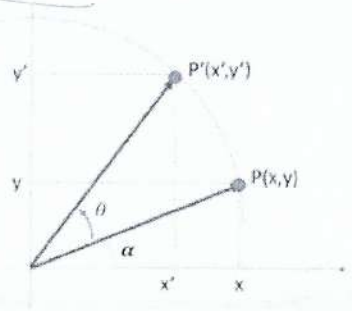
Arquivo fonte: rotaciona.{ c | cpp | java | py }

Autor: Prof. Dr. Reinaldo Arakaki(Fatec São José dos Campos)

A empresa GeoFatec trabalha com Geoprocessamento e necessita de um programa que seja capaz de rotacionar uma figura. Ou seja dado um conjunto de pontos no plano cartesiano, o programa seja capaz de rotacionar a figura de acordo com o grau dado. Sabemos que para rotacionar uma figura necessitamos de uma matriz de rotação que é dada por:

$$M = \begin{pmatrix} cos(\theta) & -sen(\theta) \\ sen(\theta) & cos(\theta) \end{pmatrix}_{2x2}$$

Onde θ é dado em graus



P'(x',y') e P(x,y) se relacionam da seguinte forma

$$A' = M.A$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix}_{2x1} = M_{2x2} \cdot \begin{pmatrix} x \\ y \end{pmatrix}_{2x1}$$

Dado um conjunto de pontos encontre as novas coordenadas dos pontos. Considerar pi=3,1415

Entrada

Uma linha contendo um número inteiro N ($1 \leq N \leq 100$) e um ângulo θ ($0 \leq \theta \leq 180$) em graus que representam o número de pontos da figura e o quanto se quer rotacionar a figura e a seguir as N coordenadas da figura (x,y) ($-1000 \leq x, y \leq 1000$)

Saída

Imprima as novas coordenadas (x,y) da figura rotacionada com duas casas decimais

Exemplo de Entrada 1	Exemplo de Saída 1
5 40 20 50 10.50 35 75 -5 -7 -2 4	-16.82 51.16 -24.48 44.73 -21.40 79.95 0.67 -8.58 -4.10 1.78

Exemplo de Entrada 2	Exemplo de Saída 2
3 100 20 250 35 4 66 50	-249.68 -23.70 -10.02 33.77 -60.70 56.32

Exemplo de Entrada 3	Exemplo de Saída 3
2 30 100 5 55 35	84.10 54.33 30.13 57.81

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema E

Grupos de apoio tático voluntário

Arquivo fonte: apoio.{ c | cpp | java | py }
Autor: Prof Antonio Cesar de Barros Munari (Fatec Sorocaba)

Preocupado com os recentes e trágicos eventos climáticos que causaram tantos danos e vítimas nos últimos meses em várias regiões do país, o prefeito de uma cidade paulista planeja criar grupos de voluntários que possam ser mobilizados para auxiliar em casos de inundações, vendavais e deslizamentos de terra mais sérios. Sua ideia é instituir esses tais "grupos de apoio tático voluntário" de maneira que possam ser treinados pela Defesa Civil para agilizar determinadas ações requeridas pelo atendimento aos cidadãos afetados pelos eventos. Um grupo de apoio desses combinaria pessoas com determinados perfis específicos estipulados pelos órgãos de planejamento municipal, como por exemplo habilidades na área da saúde (médicos, enfermeiros, paramédicos), resgate (bombeiros e profissionais com treinamento na área da segurança), acolhimento (pessoal capacitado em psicologia, serviço social), etc. Em um primeiro momento serão definidos os perfis requeridos e a quantidade de profissionais de cada perfil necessária para compor um grupo. Em seguida, após uma campanha de mobilização nos meios de comunicação, os voluntários se cadastram em um site produzido pela prefeitura e então se analisará os dados dos inscritos para decidir quantos grupos poderão ser formados em cada região do município, considerando os requisitos e a disponibilidade de voluntários. Sua tarefa será construir o programa que, dados os requisitos dos perfis e as quantidades dos voluntários em cada região, definirá quantos grupos de apoio poderão ser criados. Por exemplo, suponhamos que temos 4 perfis identificados como necessários pela prefeitura, onde o primeiro requer 2 pessoas para formar um grupo, o segundo demanda 1 pessoa, o terceiro precisa de 3 e o último tem que ter 1 pessoa. Agora suponha que uma região apresentou 7 voluntários com o primeiro perfil, 9 com o segundo, 8 com o terceiro perfil e 4 com o último perfil. Considerando os requisitos e as disponibilidades de voluntários, será possível formar apenas 2 grupos de apoio nessa região.

Entrada

A entrada se inicia com um valor inteiro Q $(0 < Q \leq 15)$, que indica a quantidade de perfis distintos requeridos para a composição de um grupo. Seguem-se então Q inteiros P_i $(0 < P \leq 30; 0 < i \leq Q)$, onde o i-ésimo valor de P indica a quantidade de pessoas requeridas com o i-ésimo perfil. A entrada continua com um inteiro V $(0 < V \leq 50)$ que indica a quantidade de regiões do município que tiveram voluntários cadastrados. Em seguida temos V linhas, cada uma descrevendo as quantidades de voluntários de cada região. Uma linha dessas se inicia com uma string R composta por uma combinação de até 10 letras e dígitos numéricos (sem espaços em branco ou outros caracteres especiais) referente ao código da região, seguida de Q inteiros D_i $(0 \leq D \leq 300; 0 < i \leq Q)$ indicando a quantidade de voluntários que possuem o i-ésimo perfil.

Saída

O programa deverá imprimir V linhas, cada uma contendo o código da região e um inteiro expressando a quantidade de grupos de apoio tático voluntário que poderão ser formados ali. Um espaço em branco deve separar esses dois itens de dado em cada linha da saída.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Exemplo de Entrada 1	Exemplo de Saída 1
4 2 1 3 1 3 ABC01 5 3 8 1 DRG32 7 9 8 4 O2932 3 3 2 9	ABC01 1 DRG32 2 O2932 0

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema F

Valida Placas

Arquivo fonte: validaplaca.{ c | cpp | java | py }
Autor: Prof Antonio Cesar de Barros Munari (Fatec Sorocaba)

Josélio foi contratado como estagiário em uma empresa que presta serviços para um órgão público que cuida, entre outras coisas, do tráfego de veículos nas vias de trânsito de seu estado. E nosso amigo Josélio é um tipo meio orgulhoso, que gosta de contar vantagem sobre seus colegas de turma na faculdade. Depois que conseguiu esse estágio então, o cara está impossível, parece que é o salvador da pátria lá no setor dele e que sem ele nada lá funciona. Recentemente ele veio contando sobre um programa que teve que fazer para validar placas de automóveis captadas por sensores óticos. Quando seu grupo de amigos falou que ele estava sendo muito chato, ele desafiou os colegas a fazerem um programa como o que ele fez. Falou que se alguém conseguisse produzir um validador que funcionasse, ele pagaria lanche para a turma inteira. Você, obviamente, se viu na situação de calar a boca desse infeliz e, ainda por cima, fazer uma merenda nas costas dele. Então vamos aos detalhes. O Brasil teve, ao longo do tempo, várias especificações para as placas de veículos automotores e, neste problema, estamos interessados apenas nas placas de automóveis. O primeiro modelo de algum interesse vigorou entre 1915 e 1941 e determinava que as placas teriam uma letra inicial seguida de uma sequência de 1 a 5 algarismos numéricos. A letra inicial poderia ser "A", para automóveis de "aluguel" tais como táxis, caminhões e ônibus, ou então deveria ser "P", indicando que tratava-se de um veículo "particular". Seriam válidas, por exemplo, as placas "A234" ou "P12345", mas não seriam válidas placas como "123X45" ou "ABCD1" ou, ainda, "B234". Em um segundo momento, no período entre 1941 e 1969, as placas passaram a ser compostas apenas por uma sequência de até 7 dígitos numéricos, como "533" ou "3949573". Uma posterior reformulação nas regras de trânsito implantou o modelo de placa veicular composta por duas letras seguidas de quatro dígitos numéricos, vigorando entre 1970 e 1990. São placas válidas neste formato, por exemplo, "RX3429" e "CM9382". Entre 1990 e 2018 o formato de placas mudou novamente, passando a ser composto por três caracteres alfabéticos seguidos de quatro dígitos numéricos, como por exemplo "BTP3465" e "PWS9521". Finalmente passamos a utilizar a chamada "placa Mercosul", que possui uma sequência inicial de três letras, seguidas por um dígito numérico, depois mais uma letra e finaliza com uma sequência de dois dígitos numéricos. O programa a ser desenvolvido por você para calar a boca do Josélio vai receber uma leitura do sensor na forma de uma string com até 50 caracteres de comprimento, contendo apenas letras em maiúsculas e dígitos numéricos. Nenhum outro tipo de caracter será encontrado nessa string, pois a coleta de dados já faz uma limpeza no conteúdo, removendo espaços em branco e eventuais pontuações presentes no registro. O programa deverá, para cada placa lida, indicar se ela é uma "Placa muito antiga" por atender ao formato vigente entre 1915 e 1941, ou se é uma "Placa numérica", caso se enquadre no formato que vigorou entre 1941 e 1969, ou, se trata-se de uma "Placa AA-9999" que teve validade entre 1970 e 1990 ou, ainda, se for uma "Placa AAA-9999" que foi exigida entre 1990 e 2018 ou, finalmente, se é uma "Placa Mercosul". Se a placa lida não for nenhuma dessas, o programa deverá informar que trata-se de uma "Placa invalida".

Entrada

A entrada possui apenas uma *string* de até 50 caracteres úteis, composta por alguma combinação de letras maiúsculas e dígitos numéricos.

Saída

Para a placa lida o programa deverá imprimir uma das respostas previstas, conforme ilustram os casos de teste. As saídas não utilizam qualquer tipo de acentuação e devem ser impressas exatamente como constam

nos exemplos.

Exemplo de Entrada 1	Exemplo de Saída 1
A1234	Placa muito antiga

Exemplo de Entrada 2	Exemplo de Saída 2
12345	Placa numerica

Exemplo de Entrada 3	Exemplo de Saída 3
AB1234	Placa AA-9999

Exemplo de Entrada 4	Exemplo de Saída 4
ABC1234	Placa AAA-9999

Exemplo de Entrada 5	Exemplo de Saída 5
ABC1D34	Placa Mercosul

Exemplo de Entrada 6	Exemplo de Saída 6
X0S0X0X0009	Placa invalida

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema G

Plantão de Dúvidas

Arquivo fonte: plantaoduvidas.{ c | cpp | java | py }

Autor: Prof. Lucio Nunes de Lira (Fatec Diadema e Fatec Ferraz de Vasconcelos)

Megan é uma excelente aluna, tanto que além de cumprir suas tarefas adequadamente também tenta ajudar seus colegas nas aulas de programação. Aliás, tomara que os colegas reconheçam isso e a agradeçam no final do semestre! Sugestão: comprem uma pizza para ela.

Depois de tantos apoios, Megan percebeu que poderia oferecer alguns plantões de dúvidas à turma, de modo que estaria disposta a fazer algumas reuniões com aqueles que solicitassem sua ajuda. Todos da turma ficaram bem empolgados e até começaram uma "vaquinha coletiva"para comprar a tal da pizza.

Entretanto, a jovem altruísta percebeu que precisaria organizar a forma como o plantão de dúvidas seria conduzido, para que seus colegas pudessem ser atendidos eficientemente. Assim, estabeleceu que quem tivesse interesse em participar precisaria fazer uma inscrição prévia, em que o aluno colocaria o nome e as categorias em que suas dúvidas se encaixam.

Com a categorização, será mais simples agrupar dúvidas semelhantes de vários alunos e chamá-los para o mesmo horário dentro do plantão de dúvidas, pois a resposta para a dúvida de um poderia ajudar o outro. Essas são as categorias disponíveis:

1. Algoritmos;
2. Boas práticas;
3. Desempenho;
4. Fluxogramas;
5. Interpretação de enunciados;
6. Sintaxe da linguagem.

Alguns dias antes de cada plantão de dúvidas, Megan disponibilizará um formulário de inscrição e, após o encerramento do prazo, divulgará uma postagem com a relação de atendimentos, que é um relatório com todas as categorias de dúvidas e os nomes dos alunos que as indicaram no formulário e que serão atendidos. Para facilitar a conferência dos atendimentos, os nomes dos alunos serão exibidos em ordem alfabética em cada categoria.

Cada colega realizará uma única inscrição, porém poderá indicar quantas categorias quiser, sendo que a inscrição sempre é feita começando com o nome do aluno seguido pelos números das categorias em que ele deseja atendimento. É importante notar que não há ordem definida nas categorias que são informadas por um aluno. Por exemplo, não há garantia de que um aluno que precisa de ajuda nas categorias 2, 3 e 6, coloque as categorias nessa ordem, ele poderia inserir 6, 2 e 3 ou qualquer outra permutação.

Como Megan ainda é humana, ela possui uma característica comum a esses seres, algo chamado 'cansaço'. Por isso, Megan indicará antecipadamente o limite de vagas para o próximo plantão de dúvidas, sendo que se a quantidade de inscrições ultrapassar esse limite, os excedentes não serão atendidos e ficarão na última categoria do relatório: 'FICA PARA PRÓXIMA!'. Neste caso, esses inscritos não atendidos serão dispostos

não em ordem alfabética, mas sim na ordem de inscrição. O preenchimento das vagas será por ordem de inscrição e a categoria 'FICA PARA A PRÓXIMA!' só deverá aparecer no relatório se houver alguém que não será atendido.

Você gostou da iniciativa de Megan e quer ajudá-la nessa aventura! Como suas habilidades em programação são lendárias, você prometeu desenvolver um programa que: (I) coletará o limite de vagas, (II) as inscrições dos alunos, na ordem em que foram feitas, um aluno por linha e (III) irá gerar um relatório no formato que ela deseja. Contamos com você!

Entrada

A primeira linha da entrada conterá um único número natural V ($V > 0$) que indicará o limite de vagas para o próximo plantão de dúvidas. As próximas linhas conterão as inscrições dos colegas de Megan que estarão sempre no formato $NA\ CD\ [CD\ CD\ \ldots]$, em que NA é o nome do aluno e cada CD a categoria da dúvida do respectivo aluno, podendo ser no mínimo um CD e no máximo seis, não há CD repetidos na mesma inscrição e todos estão no intervalo $[1..6]$. Note que NA é uma `string` composta apenas de caracteres do alfabeto latino minúsculo e sem acentuação, todos alunos farão a inscrição usando apenas o primeiro nome e há garantia que não existirão nomes repetidos. A quantidade de inscrições é, a princípio, indeterminada, isto é, o programa deverá solicitar novas entradas até que não existam mais (EOF), porém o relatório de atendimento conterá apenas os V primeiros inscritos.

Saída

A saída deverá ser conforme os casos de teste de exemplo, em que cada categoria é exibida em letras maiúsculas do alfabeto latino, sem acentuação e seguida pelos nomes dos alunos que serão atendidos e indicaram ter dúvidas nela. Note que na linha anterior e na linha posterior ao nome de cada categoria existem trinta hifens ($'-'$) e que entre as categorias há uma única linha em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Exemplo de Entrada 1	Exemplo de Saída 1
5 joao 1 2 3 4 maria 5 4 3 2 jose 5 1 carlos 5 2 1 6 4 ana 5	---------- ALGORITMOS ---------- carlos joao jose ---------- BOAS PRATICAS ---------- carlos joao maria ---------- DESEMPENHO ---------- joao maria ---------- FLUXOGRAMAS ---------- carlos joao maria ---------- INTERPRETACAO DE ENUNCIADOS ---------- ana carlos jose maria ---------- SINTAXE DA LINGUAGEM ---------- carlos

MARATONA DE PROGRAMAÇÃO
InterFatecs

Exemplo de Entrada 2	Exemplo de Saída 2
2 joao 1 maria 1 jose 1 carlos 1 ana 1	---------- ALGORITMOS ---------- joao maria ---------- BOAS PRATICAS ---------- ---------- DESEMPENHO ---------- ---------- FLUXOGRAMAS ---------- ---------- INTERPRETACAO DE ENUNCIADOS ---------- ---------- SINTAXE DA LINGUAGEM ---------- ---------- FICA PARA A PROXIMA! ---------- jose carlos ana

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema H

Quem é o pivô?

Arquivo fonte: pivo.{ c | cpp | java | py }
Autor: Prof Antonio Cesar de Barros Munari (Fatec Sorocaba)

Zequinha está estudando métodos de ordenação na escola e ficou muito empolgado com o Quick Sort, devido à sua rapidez. Fez até uma implementação em sua linguagem de programação favorita. Pesquisando um pouco mais sobre os detalhes de implementação, ficou sabendo que a escolha do valor de referência utilizado para processar uma partição dos dados, o famoso pivô, é um aspecto crítico para o desempenho. Se uma implementação descuidada do método tiver o azar de encontrar sequências particularmente ruins de pivôs em uma ordenação, o desempenho será degradado a ponto de, no limite, o Quick Sort se comparar com o Bubble Sort. Claro que isso seria um caso extremo para alguém muito azarado, mas esse alerta foi suficiente para que Zequinha resolvesse melhorar sua escolha de pivôs. Dentre várias abordagens possíveis, ele se interessou pela que escolhe três valores contidos na partição a ser processada e seleciona como pivô aquele valor que não seja sozinho nem o menor nem o maior dos três. Por exemplo, entre os valores 23, 42 e 37, o pivô a ser escolhido por esse método seria o 37; para os valores 15, 30 e 15, o escolhido teria que ser o 15 mesmo, dada a repetição. Sua tarefa é ajudar seu amigo Zequinha a determinar o valor do pivô com base em 3 valores inteiros fornecidos para análise.

Entrada

A entrada é composta por 3 inteiros separados por espaço em branco. Cada valor encontra-se na faixa que vai de -2000000000 até + 2000000000.

Saída

Imprimir o valor selecionado como pivô pelo critério exposto anteriormente.

Exemplo de Entrada 1	Exemplo de Saída 1
23 42 37	37

Exemplo de Entrada 2	Exemplo de Saída 2
15 30 15	15

Exemplo de Entrada 3	Exemplo de Saída 3
10 20 30	20

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema I

Enchendo linguiça

Arquivo fonte: linguica.{ c | cpp | java | py }
Autor: Prof Antonio Cesar de Barros Munari (Fatec Sorocaba)

Gumercindo é um profissional de TI que está desenvolvendo uma ferramenta para simular um certo tipo de problema de transporte, onde determinados fluxos são canalizados entre locais específicos de uma organização. Basicamente temos um ponto de partida único, vários locais interligados de forma hierárquica e, dependendo dos valores iniciais e das características internas desses pontos intermediários, os valores vão sendo modificados. Vamos chamar esses locais ou pontos de "estações", para melhorar a sua compreensão. Temos então uma estação inicial, de onde o conjunto de fluxos se origina, a partir de parâmetros especificados por Gumercindo em sua aplicação. Essa estação inicial vai produzir dois fluxos, uma para uma primeira estação vizinha, e outro para uma segunda estação também sua vizinha. Cada uma dessas estações vizinhas, com base em suas configurações internas e na intensidade do fluxo recebido, pode produzir até dois fluxos próprios, que vão para até duas estações em sua vizinhança imediata. E assim sucessivamente, até que estações terminais são atingidas, onde o processo como um todo se completa. Devido a detalhes técnicos totalmente fora de nosso interesse, cada estação recebe um fluxo de entrada de apenas uma estação anterior, e pode ter no máximo duas estações para onde encaminhar seus fluxos de saída. Trata-se de uma configuração em camadas regulares: a camada inicial tem apenas a estação inicial; a segunda camada é composta pelas duas estações adjacentes à inicial; a terceira camada terá até quatro estações, duas para cada uma da camada anterior, e assim sucessivamente. Uma característica adicional e, até certo ponto, simplificadora da simulação produzida por Gumercindo é que as estações vão sempre completando a capacidade das camadas iniciais e apenas a camada final poderá ter menos estações do que o seu limite. Mas nesse caso, as estações existentes nessa última camada estarão sempre a preenchendo da esquerda para a direita, sem ficarem "buracos" no arranjo. A figura 1 ilustra esse tipo de configuração em uma simulação com 8 estações dispostas em 4 camadas. Observe que a camada 1 tem uma estação (a tal "estação inicial"), a camada 2 tem as 2 estações possíveis ali, a camada 3 possui as 4 estações que nela podem ser colocadas e a última camada possui apenas uma estação, e ela está conectada no seu limite esquerdo. Caso tivéssemos mais uma estação, ela seria conectada nessa camada final imediatamente ao lado da estação já existente ali. Em outras palavras, ao construir a estrutura de conexão entre as estações, o processo ocorre camada por camada e, para cada camada, o preenchimento é sempre da esquerda para a direita. Na figura 1 a informação contida em cada estação expressa o valor do fluxo que a estação recebeu da camada anterior. No caso da estação inicial esse valor corresponde ao parâmetro de inicialização utilizado por Gumercindo naquela simulação. Assim, por exemplo, a figura mostra que a estação inicial recebeu um fluxo de valor 11 no início do processo e as duas estações da camada seguinte receberam, respectivamente, fluxos de valores 7 e 3. Em outras palavras, o dado contido em cada estação indica o fluxo recebido, conforme verificado na condução do experimento.

Sempre que uma simulação ocorre, o programa de Gumercindo gera uma sequência contendo os valores que cada estação recebeu. Assim, por exemplo, para o caso da figura 1, o programa emitiria a sequência 11, 7, 3, 19, 13, 7, 17 e 5, o que é intuitivo, pois o primeiro valor é o da estação inicial, o segundo valor é o da primeira estação da segunda camada (aquela à esquerda na figura), o terceiro valor corresponde à segunda estação dessa segunda camada (aquela à direita na figura) e assim por diante. Após uma grande quantidade de simulações, Gumercindo percebeu que uma pequena parte delas produzia um resultado final onde, para qualquer estação, em qualquer camada, o fluxo recebido nunca era maior que o fluxo que ela encaminhava para a camada seguinte, como mostra a figura 2. Perceba que à medida que avançamos pelas camadas a partir da origem, o valor do fluxo recebido por uma estação nunca é menor do que o fluxo que a sua estação antecessora na camada anterior recebeu. Gumercindo chamou esse padrão de "Configuração de tipo 1".

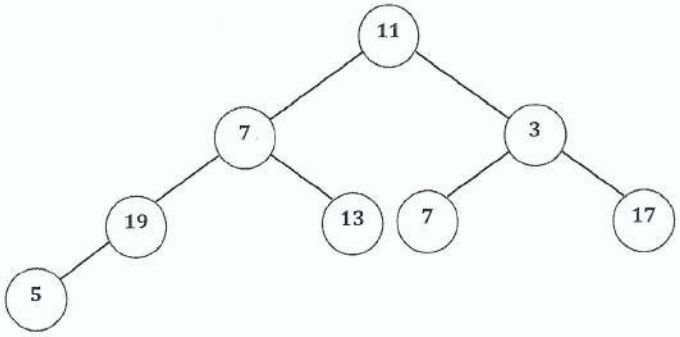


Figura 1. Representação visual de um arranjo produzido por Gumercindo.

Nesse caso verifique que o programa produziu, ao final do processamento, a sequência 3, 5, 7, 11, 13, 7, 17 e 19.

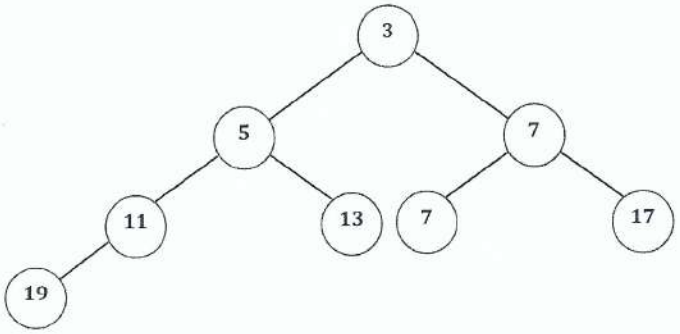


Figura 2. Representação visual de um arranjo de tipo 1 produzido por Gumercindo.

Também existia uma pequena parcela de casos onde o padrão observado era o inverso: nenhuma estação tinha um fluxo menor do que as suas sucessoras na camada seguinte. A figura 3 exemplifica esse tipo de situação, onde o programa gerou, ao seu final, a sequência 19, 13, 17, 7, 11, 3, 7 e 5. Esse tipo de arranjo recebeu o nome de "Configuração de tipo 2".

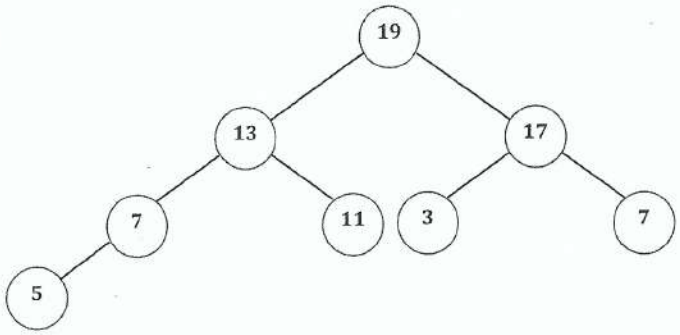


Figura 3. Representação visual de um arranjo de tipo 2 produzido por Gumercindo.

A grande maioria das simulações produziu arranjos mais parecidos com o da figura 1, onde os valores recebidos pelas estações não seguiram nenhum dos padrões anteriores. Esse caso mais comum recebeu

o nome de "Configuração de tipo 0". Cansado de analisar seus resultados manualmente e pretendendo trabalhar com quantidades maiores de camadas e estações, Gumercindo veio pedir a você, seu estagiário, para que construa um programa capaz de analisar uma sequência produzida pelo simulador e indicar se trata-se de um arranjo do tipo 0 ou do tipo 1 ou do tipo 2. Caso o arranjo possa ser classificado tanto como do tipo 1 como do tipo 2, ele deixa de ser interessante para o nosso herói, e seu programa deve classificá-lo como do tipo 0. Capriche aí no programinha, pois com ele você liberará seu chefe desse enfadonho trabalho analítico que ele chama, carinhosamente, de "encheção de linguiça" (sic).

Entrada

A entrada se inicia com um inteiro N $(0 < N \leq 1000)$ que indica a quantidade de estações existentes na simulação. Seguem-se N linhas, cada uma contendo um valor P $(-100 \leq P \leq 300)$, indicando o valor recebido por aquela estação. A sequência dos valores corresponde à sequência produzida pelo programa simulador de Gumercindo ao final do processamento, agora dispostos na forma de um valor por linha, para simplificar o seu trabalho.

Saída

Imprima um inteiro entre 0 e 2 indicando o tipo de configuração correspondente ao arranjo lido da entrada, conforme descrito anteriormente.

Exemplo de Entrada 1	Exemplo de Saída 1
8 11 7 3 19 13 7 17 5	0

Exemplo de Entrada 2	Exemplo de Saída 2
8 3 5 7 11 13 7 17 19	1

MARATONA DE PROGRAMAÇÃO
InterFatecs

Exemplo de Entrada 3	Exemplo de Saída 3
8 19 13 17 7 11 3 7 5	2

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema J
Purrinha

Arquivo fonte: purrinha.{ c | cpp | java | py }
Autor: Prof Sérgio Luiz Banin (Fatec São Paulo e Fatec São Caetano)

Purrinha é um jogo tradicional muito praticado em botecos entre goles de cerveja. Nesse jogo usam-se pequenos objetos que podem ser pedaços de papel, moedas, feijões ou palitos quebrados, ou seja, algo pequeno que possa ficar facilmente escondido dentro da mão. Vamos ficar com os palitos, por ser mais comum.

Pode ser jogado com várias pessoas.

Cada pessoa terá 3 palitos consigo e terá de escolher uma quantidade – de zero a três – a colocar na mão. Depois, todos deixam a mão fechada sobre a mesa e cada jogador deve fazer um palpite. Com esse palpite o jogador tem o objetivo de acertar o total da soma dos palitos escondidos nas mãos de todos os jogadores.

Os palpites são anunciados um por vez iniciando no jogador inicial da rodada, que pode escolher qualquer valor. Seguindo um sentido horário os jogadores farão seus palpites sucessivamente, sendo proibida a repetição dos palpites já escolhidos pelos jogadores anteriores. A cada rodada o jogador inicial é trocado (essa informação está aqui apenas para que conheçam o jogo e não deve ser levada em conta no programa).

O jogador que acertar o total de palitos ganha a rodada e com isso ganha um ponto. Ao final das rodadas o jogador que tiver o maior número de pontos é o campeão.

Por outro lado, ao final das rodadas, se dois ou mais jogadores tiverem o mesmo número de pontos, então houve empate. Escreva um programa para calcular o resultado final de um Jogo Purrinha.

Entrada

A entrada contém um caso de teste e é composta por várias linhas. Na primeira linha há um número inteiro que é a quantidade de jogadores QJ ($2 \leq QJ \leq 6$), seguida de QJ linhas contendo os nomes dos jogadores.

Em seguida, há uma linha contendo um número inteiro que é a quantidade de rodadas NR ($5 \leq NR \leq 50$), seguida de NR pares de linhas. Em cada par a linha de cima contém a mão de cada jogador e a linha de baixo contém o palpite de cada jogador. Cada uma das linhas das rodadas tem QJ números inteiros, referentes aos jogadores conforme a ordem de nomes listados na parte inicial dessa entrada de dados.

Saída

O programa deve calcular os pontos obtidos pelos jogadores e determinar quem é o vencedor ou se houve empate. Em caso de haver um vencedor imprima o nome do vencedor seguida da palavra GANHOU, com um espaço em branco entre eles. Em caso de empate imprima EMPATE. Não se esqueça do pulo de linha.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Exemplo de Entrada 1	Exemplo de Saída 1
2 LILO STITCH 3 2 0 3 2 1 0 4 1 2 1 3 1	STITCH GANHOU

Exemplo de Entrada 2	Exemplo de Saída 2
2 LILO STITCH 3 2 3 5 6 2 2 2 3 1 3 2 4	EMPATE

Exemplo de Entrada 3	Exemplo de Saída 3
3 HOUSE MONK SHELDON 6 1 2 0 6 3 5 0 3 1 2 4 5 0 0 3 6 1 8 1 3 3 5 7 9 2 0 0 2 1 6 2 3 2 5 3 7	MONK GANHOU

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema K

Mentirinha

Arquivo fonte: mentirinha.{ c | cpp | java | py }

Autor: Prof. Lucio Nunes de Lira (Fatec Diadema e Fatec Ferraz de Vasconcelos)

Thiago gosta muito de matemática, tanto que até inventa suas próprias definições sobre números! Por exemplo, Thiago é fascinado pelos números primos e já sabe diversas características sobre eles, em uma tarde chuvosa pensou que poderia criar uma definição chamada "primos de mentirinha".

Como sabemos, números naturais primos têm apenas dois divisores naturais, isto é, são divisíveis por 1 e por eles mesmos. Já os "primos de mentirinha"de Thiago têm apenas um divisor a mais, ou seja, têm três divisores naturais.

Sua missão é criar um programa que possa ajudar Thiago a descobrir se um dado número natural pode ser considerado um "primo de mentirinha".

Entrada

A entrada contém um único número natural N $(1 < N < 1000000)$ que é o número que Thiago quer saber se é um "primo de mentirinha".

Saída

A saída será a palavra 'sim' ou a palavra 'nao' (sem apóstrofos, sem acentuação e em minúsculo), que indicará se N é um "primo de mentirinha"segundo a definição de Thiago.

Exemplo de Entrada 1	Exemplo de Saída 1
7	nao

Exemplo de Entrada 2	Exemplo de Saída 2
4	sim

Exemplo de Entrada 3	Exemplo de Saída 3
9	sim

Exemplo de Entrada 4	Exemplo de Saída 4
994009	sim

Exemplo de Entrada 5	Exemplo de Saída 5
999983	nao

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problem L

Molecular scissors

Source file: molecularscissors.{ c | cpp | java | py }
Author: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

Within the universe of molecular biology, there exist special molecules called enzymes that act as catalysts for biochemical reactions. A specific family of enzymes, known as restriction enzymes, have a unique affinity for specific sequences of nucleotides in a strand of DNA. These enzymes "read"the DNA strand and cut it at their respective recognition sites, acting as molecular scissors.

Scientists have discovered a unique characteristic of the recognition sites of certain enzymes, particularly restriction enzymes. The nucleotide sequence on one strand, when read from a specific direction, matches the sequence on the complementary strand, read in the opposite direction.

For example:

Strand: AAGCTT

Complementary: TTCGAA

As shown, reading the strand from left to right is identical to reading the complementary strand from right to left.

Conversely, a strand like the following does not exhibit this characteristic:

Strand: AAGG

Complementary: TTCC

Restriction enzymes can cut sequences like AAGCTT and its complementary TTCGAA, but not sequences like AAGG and TTCC. As real examples, the restriction enzyme EcoRI recognizes the sequence GAATTC, and its complementary strand would read CTTAAG. Similarly, the enzyme HindIII recognizes the sequence AAGCTT and the complementary strand reading TTCGAA. These enzymes are crucial in molecular biology, allowing scientists to cut DNA strands at specific locations, facilitating various genetic engineering and research applications.

As a Bioinformatics intern, your task is to develop a program to assist researchers in identifying potential recognition sites (inside of a strand) for such a type of enzymes. The program should identify sequences that exhibit the aforementioned characteristic.

Input:

The input consists of multiple lines, each representing a DNA sequence, composed of the characters A, T, C, and G, representing the nucleotides Adenine, Thymine, Cytosine, and Guanine respectively. The maximum length of each DNA sequence is 10^5. The input ends with EOF.

Output:

For each DNA sequence, if no site (part of the sequence) that can be matched by a restriction enzyme is found, output "false". If such a site is found, output the starting position and the length of the longest such sequence within the DNA strand. If more than one site presents the same length, present the starting position and the length of the first one.

Example:

Input:

Example of Input 1	Example of Output 1
GAATTC AAGCTTTCGAAGCTTAAAAAA CCGGAAGGCCGG ATT AAGCTCAA AA AAGG	1 6 1 6 1 4 false 2 4 false false