INSTITUTO FEDERAL SÃO PAULO	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO**

VII MARATONA DE PROGRAMAÇÃO INTERIF 2024

Aquecimento

Caderno de Problemas

Informações Gerais

A) Sobre a entrada

1) A entrada de seu programa deve ser lida da entrada padrão.
2) A entrada é composta de um ou mais casos de teste, 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.
5) O final da entrada pode coincidir com o final do arquivo ou com uma entrada determinada

B) Sobre a saída

1) A saída de seu programa deve ser escrita na saída padrão.
2) Espaços em branco só devem ser colocados quando solicitado.
3) Cada linha, incluindo a última, deve conter o caractere final-de-linha.

C) Regras

1) Só é permitida a comunicação entre os membros de um mesmo grupo.
2) Não é permitida a comunicação com o técnico (coach) do time.
3) Eventuais dúvidas sobre a prova utilizar o menu “clarification” do sistema de submissão.

D) Ambiente computacional

O sistema de correção das submissões será executado utilizando a distribuição Ubuntu GNU/Linux 20.04.2 LTS amd64, tendo os seguintes compiladores/interpretadores configurados:

- C - gcc version 11.4.0 (Ubuntu 9.3.0-17ubuntu1~22.04)
- C++ - gcc version 11.4.0 (Ubuntu 9.3.0-17ubuntu1~22.04)
- Python 3 - Python 3.10.12
- Java - openjdk-17.0.10
- C# - mono JIT 6.12

Problema A
Coronavírus

Criado por Jones Mendonça de Souza (IFSP – campus Barretos)
Arquivo: corona.[c|cpp|cs\java\py]
***Timelimit:** 1*

Com o aparecimento dos casos de doença respiratória causada pelo coronavírus, o governo brasileiro vem adotando medidas de preparação, orientação e controle para um possível atendimento de casos suspeitos no país. A principal forma de transmissão do coronavírus se dá por contato próximo de pessoa a pessoa, e muitos não entendem as estatísticas de crescimento das pessoas infectadas. Como você é um bom programador e tem uma boa lógica de programação o governo Brasileiro pediu que você criasse um algoritmo que dado o número de infectados mostre na tela a probabilidade de pessoas que irão se infectar se ficarem próximas umas das outras. Sabe-se que: 1 infecta 2; 16 infecta 256; 65.536 infecta 4.294.967.296

Entrada

A entrada é composta por um número inteiro representando o número de pessoas que já estão infectadas.

Saída

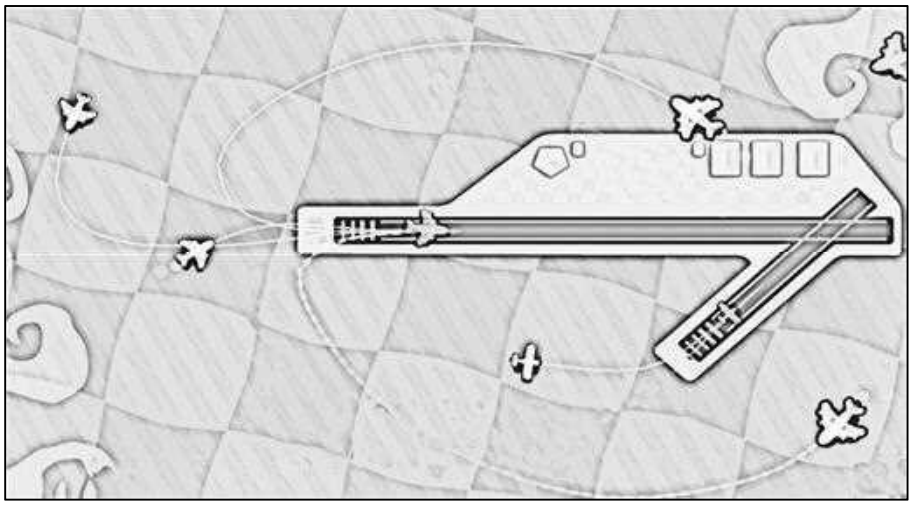
A saída deverá exibir a probabilidade de pessoas que irão de infectar se tiverem contato com outras pessoas.

Exemplos de Entradas	Exemplos de Saídas
1	2
16	256
65536	4294967296

Problema B
Controlador de Tráfego Aéreo

Criado por Jones Mendonça de Souza (IFSP – campus Barretos)
Arquivo: fly.[c|cpp|cs\java\py]
***Timelimit:** 1*

O aeroporto de Congonhas recebe todos os dias uma média de 600 pousos e decolagens, ou cerca de 36 por hora. No último ano, foram exatamente 223.989 movimentos aéreos. Para organizar todo o fluxo de aviões que chegam a Congonhas e saem de lá, a torre de controle funciona o tempo inteiro com nível máximo de atenção. Para descartar qualquer possibilidade de erro humano o chefe do controle de tráfego aéreo de Congonhas contratou a sua equipe da Maratona de Programação InterIF para desenvolver um programa que organize automaticamente o fluxo de aviões no campo de pouso. Para isso, basta seguir o seguinte protocolo, os aviões que veem do lado Oeste da pista têm maior prioridade de serem colocados na fila, pois são aqueles que estão mais próximo do localizador (início da pista). Já os aviões que estão se aproximando pelo lado Norte e Sul, devem ser inseridos na fila 1 por vez, ou seja, insere-se 1 avião do lado Norte e em seguida 1 avião do lado Sul. Por último, insere-se o próximo avião que esteja se aproximando ao lado leste da pista.



Entrada

A entrada é composta por um número inteiro P, representando o ponto cardeal do avião que entrou no campo da pista (-4 <= P <= -1), onde (-4 representa o lado leste, -3 o lado norte, -2 lado sul e -1 lado oeste) . Em seguida é realizada a entrada dos respectivos aviões, compostos de um identificador começando com a letra "A" seguida de um número inteiro I (1 <= I <= 1000). A qualquer momento é permitido trocar o ponto cardeal, e inserir novas aeronaves, repetidamente até que o controlador finalize a sessão com o dígito 0.

Saída

A saída é composta de uma linha contendo as aeronaves enfileiradas pela ordem do protocolo estabelecido pelo aeroporto. O resultado de seu programa deve ser escrito na saída padrão.

Exemplos de Entradas	Exemplos de Saídas
-4 A1 A26 A38 A23 -1 A80 A40 -2 A2 A16 A108 -3 A20 A44 0	A80 A20 A2 A1 A40 A44 A16 A26 A108 A38 A23
-4 A12 A33 -3 A8 A33 -2 A77 A102 A866 -3 A21 A15 A9 -1 A2 0	A2 A8 A77 A12 A33 A102 A33 A21 A866 A15 A9

Problema C
Biometria das Girafas

Criado por Jones Mendonça de Souza (IFSP – campus Barretos)

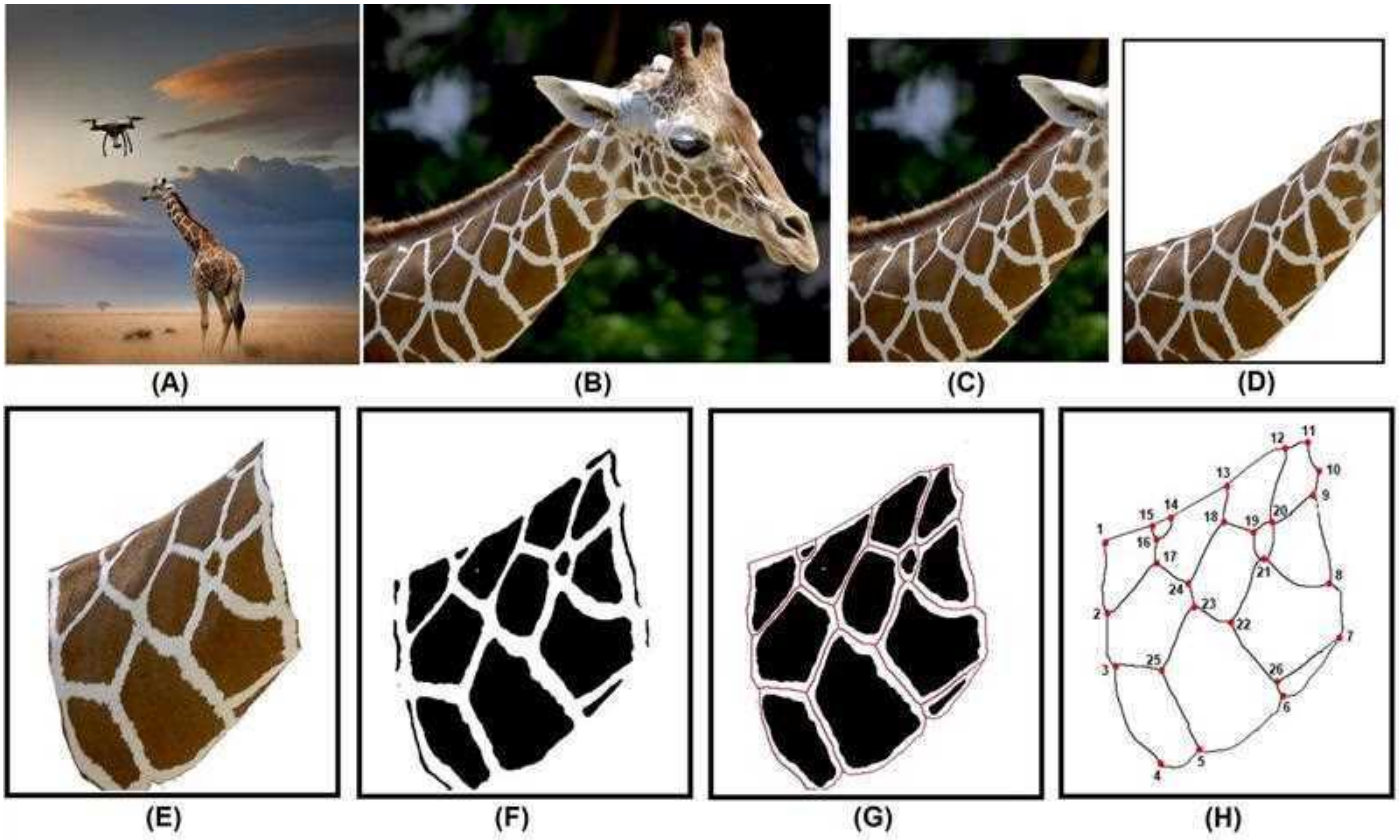
Arquivo: biometria.[c | cpp | cs | java | py]

timelimit: 1s

Um grupo de pesquisadores do IFSP se reuniu para realizar um estudo inédito sobre as girafas. Sabe-se que a diferenciação de sexo entre esses animais é realizada pelo peso, comprimento das pernas e do pescoço. No entanto, um estudo de 2020 na Alemanha revelou que essas características já não são confiáveis devido à falta de nutrientes no sul do Saara, que afeta o desenvolvimento dos machos, tornando suas características semelhantes às fêmeas.

Preocupados com a extinção, os professores Murilo Vargas, Jones Souza e Cassio Onodera, estão desenvolvendo um algoritmo usando os padrões de manchas na pele das girafas para identificar o sexo. A professora Giovana Nakashima, especialista em biologia animal, confirmou que os machos possuem 50% a menos de manchas na região central do pescoço. Para contar as manchas, Cassio criou um drone (Figura A), que obtém coordenadas e captura imagens de alta resolução da área do pescoço (Figura B). Jones desenvolveu algoritmos de processamento de imagens (Figura C, D, E, F), extração de características (Figura G) e representação de dados (Figura H). Atualmente, o projeto está parado há 1 ano com o professor Murilo, que tem em mãos os dados, mas não está conseguindo desenvolver um algoritmo que realize a contagem das manchas dos animais.

Como Murilo estará na final da VI Maratona InterIF, ele resolveu compartilhar seu problema com o grupo de alunos que estarão competindo na Maratona. Será que você consegue ajudar o professor Murilo?



Entrada

A entrada é composta por um grafo bidirecional, representando o mapeamento das manchas. A primeira linha possui dois inteiros, separados por um espaço, um inteiro **N** ($2 \leq N \leq 10^9$), que indica o número de pontos do grafo e um inteiro **M** ($1 \leq M \leq 10^9$), que indica o número de ligações entre pares de pontos. Depois, separados por um espaço, seguem-se **M** linhas com os inteiros distintos **U (1 ≤ U ≤ N)** e **V (1 ≤ V ≤ N),** separados por um espaço, indicando que existe uma ligação bidirecional entre **U** e **V.** Como pode haver erros na etapa de segmentação das características, não é garantido que todos os pontos estejam interligados.

Saída

A saída deve apresentar um único inteiro indicando a quantidade de manchas existentes na pele do animal.

Exemplos de Entradas	***Exemplos de Saídas***
4 5 1 2 2 3 2 4 3 4 3 1	2
26 38 1 2 1 15 2 3 2 17 3 25 3 4 4 5 5 6 5 25 25 23 23 24 23 22 24 18 24 17 17 16 16 15 16 14 15 14 14 13 13 12 13 18 18 19 19 20 19 21 20 12 20 9 20 21 21 22 21 8 8 9 8 7 9 10 10 11 11 12 7 26 7 6 6 26 26 22	13

PROBLEMA D
BK em Birigui

Nome do arquivo fonte: bk.[c | cpp | cs | java | py]
(Prof. Ms. Murilo Varges da Silva – IFSP - Birigui)

É apenas uma moda passageira ou está aqui para ficar? O número de lanchonetes "*fast-food*" que estão abrindo franquias em Birigui está crescendo de forma acelerada, este crescimento já despertou o interesse de duas das dez maiores redes de "*fast-food*" do mundo (*Subway e McDonald's*), que já iniciaram operação na cidade.

Diante desta tendência outros gigantes do seguimento estão de olho na cidade de Birigui, também conhecida como "A *Massachussets* Brasileira" por sediar os grandiosos jogos do lendário "Rock Gol" campeonato de futebol disputado por músicos, exibido pela MTV entre 1995 e 2008 que era apresentado por Paulo Bonfá.

Nesta briga de gigantes surge o BK - Burger King ou "*Rei dos Hambúrgueres*" que está de olho nesta oportunidade, eles estão interessados em se sediar em Birigui e precisam identificar locais onde já se concentram restaurantes deste tipo para tentar "roubar" a clientela dos concorrentes. Sua tarefa é a partir de um mapa da cidade, marcado com as localizações dos restaurantes "*fast-food*, encontrar o local mais próximo do maior número de restaurantes na cidade. Como você provavelmente já sabe, sua cidade é construída em um layout de quadras, com blocos alinhados em eixos norte-sul e leste-oeste. Deste modo você tem que caminhar ao longo das ruas, e a distância entre interseções é (a, b) e (c, d) é | a - c | + | b - d |.

Entrada

A entrada contém vários casos de teste. Cada caso de teste descreve uma cidade. A primeira linha de cada caso de teste contém quatro inteiros ***dx, dy, n,*** e ***q***. Estas são as dimensões da cidade $dx \times dy$ ($1 \leq dx, dy \leq 1000$), o número de restaurantes "*fast-food*" n ($0 \leq n \leq 5 * 10^5$), e o número de consultas q ($1 \leq q \leq 20$). Cada uma das próximas n linhas contém dois inteiros x_i e y_i ($1 \leq x_i \leq dx$, $1 \leq y_i \leq dy$); eles especificam a localização do restaurante ¡enésimo. Haverá no máximo, um restaurante por cruzamento. Cada uma das próximas q linhas contém um único inteiro m ($0 \leq m \leq 106$), indicando a distância máxima entre um restaurante e um ponto de interesse. O último caso de teste é seguido por uma linha contendo quatro zeros.

Saída

Para cada caso de teste da entrada, exibir seu número do processo. Em seguida, exibir uma linha por consulta do caso de teste. Cada linha exibe o número máximo de restaurantes alcançáveis para a consulta de determinada distância m seguido do local ideal. Por exemplo, a saída de exemplo do "Caso 1" mostra que 3 restaurantes estão dentro da distância de consulta 1 e a localização ideal é (3,4), 4 restaurantes estão dentro da distância de consulta 2 e a localização ideal é (2,2) e 5 restaurantes estão a uma distância de consulta 4 e a localização ótima é (3,1). Se houver vários locais ideais, escolher o local que está mais ao sul (o menor inteiro positivo coordenada y). Se ainda houver um empate, escolher o local mais ocidental (o menor inteiro positivo coordenada x). Siga o formato da saída do exemplo.

Exemplos:

Entradas	Saídas
4 4 5 3 1 1 1 2 3 3 4 4 2 4 1 2 4 1 1 0 3 0 1 2 1 1 1 3 1 1 0 1 2 10 10 1 4 7 7 0 1 5 10 4 4 2 5 2 3 3 2 4 3 2 1 0 0 0 0 0	Caso 1: 3 (3,4) 4 (2,2) 5 (3,1) Caso 2: 0 (1,1) 0 (1,1) 0 (1,1) Caso 3: 1 (1,1) 1 (1,1) 1 (1,1) Caso 4: 1 (7,7) 1 (7,6) 1 (7,2) 1 (3,1) Caso 5: 2 (1,1) 2 (1,1) 2 (2,1) 2 (2,2) 1 (3,2)

PROBLEMA F
Cadastro de Duendes de Nlogônia

Nome do arquivo: cdn. [c \cpp\cs\java\py]
(Prof. Ms. Murilo Varges da Silva – IFSP - Birigui)

Nlogônia é uma cidade povoada por duendes localizada na região sul da Deep Web, devido ao crescimento populacional e ao interesse do governo local de arrecadar mais impostos foi criado o CDN (Cadastro de Duendes de Nlogônia), este cadastro consiste em um número único que será gerado pelo governo para cada cidadão.

Como uma forma de evitar fraudes e falsificações o governo de Nlogônia adotou um padrão para este número que terá o seguinte formato NNN.NNN-D, sendo os seis Ns o número do documento e D o dígito verificador, sua tarefa é desenvolver um programa que valide o número do CDN e informe se o documento é válido ou não, seguindo a seguinte regra:

O número de um CDN tem 6 algarismos e mais um dígito verificador, que é indicado após um traço. Logo, um CDN tem 7 algarismos. O número do CDN é formatado da seguinte forma ABC.DEF-G, logo o CDN formatado contém 9 dígitos, onde os algarismos não **podem ser todos iguais** entre si. O **G** é chamado de dígito verificador do número do CDN.

Cálculo Dígito Verificador

Para obter G multiplicamos A, B, C, D, E e F pelas constantes correspondentes:

Elemento CDN	A	B	C	D	E	F
Multiplicador	20	19	17	14	10	5

O resultado da soma, 20*A + 19*B + 17*C + 14*D + 10*E + 5*F, é dividido por 11.
Analisamos então o RESTO dessa divisão:
Se for 0 ou 1, o dígito **G** é [0] (zero). Se for 2, 3, 4, 5, 6, 7, 8, 9 ou 10, o dígito **G** é [11 - RESTO].

Exemplo de Cálculo Dígito Verificador
CDN: 041.467-9

Elemento CDN	A	B	C	D	E	F
Valor CDN	0	4	1	4	6	7
Multiplicador	20	19	17	14	10	5

O resultado da soma, 20*0 + 19*4 + 17*1 + 14*4 + 10*6 + 5*7 é **244.**
O resto da divisão 244 por 11 é **2.**
Como o resto não é 0 ou 1 o **cálculo do digito** é feito utilizando a seguinte fórmula **11 - RESTO**, portanto **11-2 = 9**, neste caso o CDN 041.467-**9** é válido.

Entrada

A primeira linha da entrada contém um número inteiro n ($1 \leq n \leq 1000$), que indica o número de documentos que serão validados.

As n linhas seguintes contém cada uma delas os documentos a serem validados.

Saída

Para cada documento (CDN) deve-se apresentar se o documento é válido ou não.

Exemplos:

Entradas	Saídas
12 041.467-9 169.724-4 962.464-5 827.961-5 827.436-4 222.222-2 153.292-3 716.718-7 771.538-2 299.035-0 811.322-3 111.111-1	CDN VALIDO! CDN VALIDO! CDN INVALIDO! CDN INVALIDO! CDN INVALIDO! CDN INVALIDO! CDN VALIDO! CDN INVALIDO! CDN INVALIDO! CDN VALIDO! CDN INVALIDO! CDN INVALIDO!

25 041.467-9 169.724-4 962.464-5 827.961-5 827.436-4 153.292-3 716.718-7 771.538-2 299.035-0 811.322-3 141.711-8 644.662-7 723.741-8 035.190-7 106.040-3 648.446-0 370.350-1 393.548-3 954.756-7 376.931-6 439.626-7 118.082-1 115.639-3 930.977-5 386.021-4	CDN VALIDO! CDN VALIDO! CDN INVALIDO! CDN INVALIDO! CDN INVALIDO! CDN VALIDO! CDN INVALIDO! CDN INVALIDO! CDN VALIDO! CDN INVALIDO! CDN INVALIDO! CDN INVALIDO! CDN INVALIDO! CDN VALIDO! CDN VALIDO! CDN INVALIDO! CDN VALIDO! CDN INVALIDO! CDN VALIDO! CDN VALIDO! CDN INVALIDO! CDN INVALIDO! CDN VALIDO! CDN VALIDO! CDN INVALIDO!

5 962.464-5 041.467-9 827.961-5 169.724-4 827.436-4	CDN INVALIDO! CDN VALIDO! CDN INVALIDO! CDN VALIDO! CDN INVALIDO!

Problem F
Caça ao Tesouro

Adaptado por Murilo Varges da Silva (IFSP – campus Birigui)
Arquivo: tesouro.[c|cpp|cs|java|py]
***Timelimit:** 1*

O famoso pirata "Capitão Jack Sparrow" foi um renomado astrônomo e matemático. Ele enterrou a maioria de seus tesouros na ilha caribenha de São Basil, onde o Pico Colombo é um conhecido marco geográfico. Jack Sparrow desapareceu quando sua frota de três navios foi pega em um furacão em 1582. Talvez por algum tipo de premonição, antes de sua excursão fatal, ele escreveu em uma carta para uma de suas sobrinhas na Suécia a distância exata ao seu tesouro oculto, partindo Pico Colombo em direção sul.

Preocupado que seu mapa pudesse acabar nas mãos erradas, Jack Sparrow usou suas habilidades em matemática como seguro contra ladrões. Em vez de escrever na carta o número indicando a distância, ele multiplicou-o por um segundo número N, e escreveu o resultado D na carta, junto com o valor de N e uma explicação de como o cálculo deveria ser feito. Ele sabia que mesmo se uma pessoa indesejada obtivesse a carta, ela deveria saber como dividir dois números, coisa que poucos criminosos conseguiam fazer naquele tempo. Infelizmente, quando a carta chegou em seu destino na Europa, a sobrinha Jack Sparrow havia entrado em um convento e nem se importou em abrir a carta.

Exatamente quatro séculos após o ocorrido, Maria por ventura veio a obter um baú com os pertences de sua ancestral freira. E você pode imaginar sua surpresa quando ela descobriu a carta, ainda lacrada! Maria está planejando uma viagem para buscar o tesouro de Jack Sparrow, mas ela precisa de sua ajuda. Apesar do valor de N estar intacto e ela poder lê-lo, o número D foi parcialmente comido por traças de forma que apenas alguns dos dígitos estão visíveis. A única pista que Maria tem é que o digito mais à esquerda de D não é zero pois Jack Sparrow disse em sua carta.

Dada a representação parcial de D e o valor de N, você deve determinar o menor valor possível de D de forma que este seja um múltiplo de N e que não comece com zeros.

Entrada

A entrada consiste de uma única linha que contém uma string S não vazia com no máximo 1000 caracteres e um inteiro N ($1 \leq N \leq 1000$). Cada caractere de S é ou um digito ou o caractere "?" (question mark); o digito mais à esquerda não é "0" e no mínimo um caractere de S é "?".

Saída

Imprima uma única linha com um inteiro D, que não comece com zeros, indicando o menor múltiplo de N que possua |S| dígitos e cujos dígitos em S coincidam com os dígitos correspondentes em D. Caso não exista tal inteiro D, imprima um "*" (asterisco) para a saída.

Exemplos de Entradas	Exemplos de Saídas
1?????????????????????????????? 2	100000000000000000000000000000
??????????????????????????????1 2	*
?294?? 17	129404

Problema G

Triângulos

Adaptado ACM/ICPC por Jones Mendonça de Souza – IFSP - Barretos

Arquivo: triangulos.[c|cpp|cs|java\py]
***Timelimit:** 6*

Para a sua viagem a Pequim, você trouxe muitos livros de quebra-cabeças, muitos deles contendo desafios como o seguinte: quantos triângulos podem ser encontrados na Figura I ?

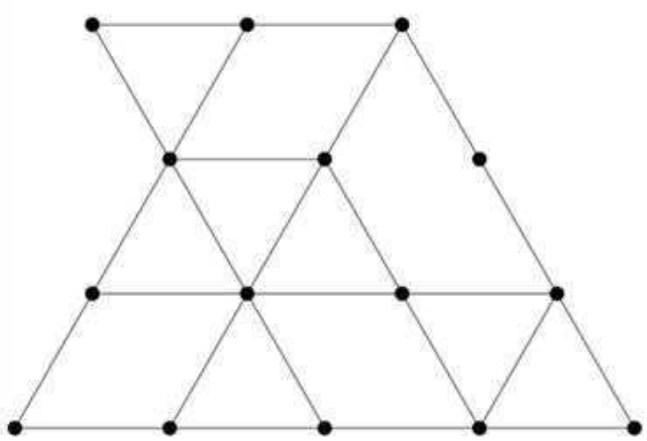


Figura I - Ilustração do exemplo de entrada 2

Enquanto esses quebra-cabeças mantêm o seu interesse por um tempo, você rapidamente fica entediado com eles e começa pensando em como você pode resolvê-los por algoritmos. Quem sabe, talvez um problema como esse realmente pode ser proposto na maratona de programação deste ano. Bem, adivinhe? Hoje é seu dia de sorte!

Entrada

A primeira linha de entrada contém dois inteiros r e c ($1 \leq r \leq 3.000$, $1 \leq c \leq 6.000$), especificando o tamanho da imagem, em que r é o número de linhas dos vértices e c é o número de colunas. Depois disso, 2r-1 linhas, cada uma com no máximo 2c-1 caracteres. Linhas ímpares contêm vértices (representados como caracteres) e zero ou mais bordas horizontais, enquanto as linhas pares contêm zero ou mais bordas diagonais. Especificamente, linhas de imagens com números 4k + 1 possuem vértices nas posições 1,5,9,13, ... enquanto as linhas com números 4k + 3 têm vértices nas posições 3,7,11,15, ... Todos os possíveis vértices são representados na entrada (por exemplo, veja como é representada a entrada 2 no exemplo da

Figura I). As arestas horizontais que conectam os vértices vizinhos são representadas por três traços. Bordas diagonais são representadas por um único caractere de barra ('/') ou barra invertida ('\'). Os caracteres de borda serão colocados exatamente entre os vértices correspondentes. Todos os outros caracteres serão caracteres espaciais. Observe que, se qualquer linha de entrada puder conter espaço em branco à direita, esse espaço em branco poderá ser omitido.

Saída

Exiba o número de triângulos (de qualquer tamanho) formados pelas bordas da grade na imagem de entrada.

Entrada 1	Saída 1
3 3 x---x \ / x / \ x x	1

Entrada 2	Saída 2
4 10 x x---x---x x \ / / \ x x---x x x / \ / \ \ x x---x---x---x / / \ \ / \ x---x---x---x---x	12

Problema H
Futebol na TV

Nome do arquivo: futebol. [c | cpp | cs | java | py]
(Prof. Ms. Murilo Varges da Silva – IFSP - Birigui)

Marcongekson adora assistir jogos de futebol na TV, entretanto ultimamente o nível dos jogos não tem sido dos melhores. Ele está se preparando para começar a assistir um jogo hoje, o jogo dura 90 minutos e não possui intervalos.

Cada minuto pode ser interessante ou chato. Se 15 minutos consecutivos são chatos, Marcongekson em seguida, desliga a TV.

Você sabe que haverá ***n*** minutos interessantes t1, t2, ..., tn. Sua tarefa é calcular por quantos minutos Marcongekson vai assistir ao jogo.

Entrada

A primeira linha da entrada contém um número inteiro N ($1 \leq N \leq 90$), que indica o número de minutos interessantes.

A segunda linha contém ***n*** inteiros t1, t2, ..., tn ($1 \leq t1 < t2 < ... \ tn \leq 90$) indicando os minutos interessantes, digitados na ordem crescente.

Saída

Imprimir o número de minutos que Marcongekson vai assistir ao jogo.

Exemplos:

Entradas	**Saídas**
3 7 20 88	35

9 16 20 30 40 50 60 70 80 90	15

9 15 20 30 40 50 60 70 80 90	90

PROBLEMA I
Bônus do Funcionário

Nome do arquivo fonte: bonus. [c | cpp | cs | java | py]
(Autor: Prof. Cássio Agnaldo Onodera e Valtemir de Alencar e Silva)

Antônio trabalha no departamento pessoal da prefeitura de uma cidade da Nlogônia. Nlogônia é um país recém-emancipado localizado no leste asiático. O prefeito, empossado recentemente, resolveu agradar os funcionários da prefeitura e concederá um bônus em dinheiro baseado na idade em dias da pessoa. Cada funcionário receberá N$ 2,00 (dois nlogs) por dia de vida. O cálculo dos dias de vida deve ser baseado na data de 10/06/2015. Considere que cada ano possua 360 dias e cada mês possua 30 dias.

Sua tarefa é ajudar Antônio a realizar este cálculo, desenvolvendo um programa de computador que receba a data de nascimento do funcionário (dia, mês e ano) e mostre o valor em nlogs que o funcionário receberá.

Entrada

A entrada contém um único caso de teste com uma única linha com três valores: D, M e A; onde D é o dia de nascimento do funcionário ($1 \leq D \leq 30$), M é o mês de nascimento do funcionário ($1 \leq M \leq 12$) e A é o ano de nascimento do funcionário ($1900 \leq A \leq 2015$).

Saída

A saída será uma única linha contendo o valor inteiro em nlogs (N$) que o funcionário receberá.

Exemplo 1:

Entrada	Saída
10 06 2014	720

Exemplo 2:

Entrada	Saída
13 02 1969	33354

Exemplo 3:

Entrada	Saída
09 06 2015	2

PROBLEMA J
Truco

Nome do arquivo fonte: truco. [c | cpp | cs | java | py]
(Prof. Ms. Cássio Agnaldo Onodera – IFSP - Birigui)

Natália iniciou seu curso de Engenharia Química neste ano e fez muitas amizades em sua faculdade. Como na maioria das faculdades e universidades o jogo de cartas mais disputado entre os alunos é o truco.

Como Natália nunca havia jogado truco, ela possui alguma dificuldade em identificar se uma carta é mais forte que outra.

No truco, a sequência das cartas da mais fraca para a mais forte é: 4, 5, 6, 7, 11 (Dama ou "Q"), 12 (Valete ou "J"), 13 (Reis ou "K"), 1 (Ás ou "A"), 2 e 3. Ou seja, a carta número 4 é a mais fraca de todas e a carta "3" é a mais forte. Isto, desconsiderando as "manilhas", que são cartas mais fortes que a carta de número 3. No truco, as cartas 8, 9 e 10 devem ser removidas do baralho.

Desconsiderando as "manilhas", escreva um programa que auxilie Natália a identificar se uma carta é mais forte que a outra.

Entrada

A entrada contém vários casos de teste. Cada caso de teste receberá dois valores V1 ($1 \leq V1 \leq 13$ – excluindo o 8, 9, e 10) e V2 ($1 \leq V2 \leq 13$ – excluindo o 8, 9, e 10) que simbolizam os valores das cartas. Estes valores serão digitados em uma única linha separados por um espaço em branco. O final da entrada é indicado por uma linha que contém apenas números zero.

Saída

Para cada caso de teste, a saída deverá imprimir uma palavra informando se a primeira carta é mais forte que a segunda. De acordo com a análise dos valores da carta o programa deverá imprimir uma única saída contendo uma das palavras: "MAIOR", "MENOR" ou "IGUAL". Se a primeira carta for mais forte que a segunda, deverá imprimir a palavra "MAIOR"; se a segunda carta for mais forte que a primeira, deverá imprimir a palavra "MENOR" e se nenhuma das opções for verdadeira, imprimir a palavra "IGUAL".

Exemplos:

Entradas	Saídas
11 7 1 13 7 3 2 3 5 5 0 0	MAIOR MAIOR MENOR MENOR IGUAL

PROBLEMA K
Latinhas

Nome do arquivo fonte: latinhas. [c | cpp | cs | java | py]
(Autor: Prof. Ms. Cássio Agnaldo Onodera- IFSP -Birigui)

A professora Aline leciona em uma escola municipal e para conscientizar os alunos da importância da reciclagem de materiais para a preservação do meio ambiente, resolveu organizar uma gincana com os alunos para coletar latinhas de alumínio que serão encaminhadas para a reciclagem. Nessa gincana os alunos terão que levar as latinhas coletadas até a escola que serão contadas e atribuídas à sala de aula que o aluno estuda. No final, cada classe terá uma pontuação correspondente ao número de latinhas coletadas. A classe que tiver a maior pontuação será a vencedora e ganhará um passeio até o zoológico municipal.

Como a professora Aline não entende nada de programação, ela solicitou à você que desenvolvesse um programa de computador que recebesse a pontuação obtida por cada sala de aula e no final informasse qual foi a sala vencedora.

Entrada

A entrada contém vários casos de teste, representando cada gincana. A primeira linha de um caso de teste contém um número inteiro N (1 <= N <= 1000) que representa o número de salas participante da gincana. As próximas N linhas contém um inteiro L (0 <= L <= 10000) indicando a pontuação obtida pela sala. O primeiro valor L, representa a pontuação da sala 1, o segundo valor L, representa a pontuação 2 e assim sucessivamente. O final da entrada é indicado por uma entrada N=0.

Os dados de entrada devem ser lidos da entrada padrão.

Saída

Para cada caso de teste, seu programa deve imprimir uma única linha, contendo um número inteiro representando o número da sala vencedora.

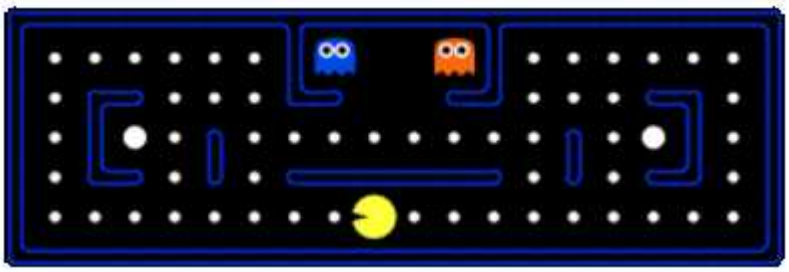
O resultado deve ser escrito na saída padrão.

Exemplos:

Entrada	Saída
3 3456 2629 7930 4 1221 2922 1022 982 0	3 2

PROBLEMA L
Pacman

Arquivo: pacman. [c | cpp | cs | java | p y]
(Prof. Ms. Cássio Agnaldo Onodera – IFSP - Birigui)



"Pac-man ou ***Pacman*** é um jogo eletrônico criado por Tôru Iwatani para a empresa Namco nos início dos anos 1980, tornou-se um dos jogos mais populares, tendo versões para diversos consoles. A mecânica do jogo é simples: o jogador era uma cabeça redonda com uma boca que se abre e fecha, posicionado em um labirinto simples repleto de pastilhas e 4 fantasmas que o perseguiam. O objetivo era comer todas as pastilhas sem ser alcançado pelos fantasmas, em ritmo progressivo de dificuldade" (*Wikipedia*).

Seu objetivo é criar uma releitura deste famoso jogo dos anos 1980. O tabuleiro será formado por um retângulo com L linhas e C colunas. Inicialmente o tabuleiro estará preenchido com pastilhas simbolizadas por asteriscos ("*"). Em alguns pontos do tabuleiro se encontram fantasmas simbolizados por uma letra efe ("F"). O personagem (Pacman), simbolizado por uma letra pê ("P"), será controlado pelo usuário. Quando o personagem passa por uma posição que se encontra uma pastilha, o jogador ganha 1 (um) ponto e ao encontrar um fantasma, a pontuação obtida até o momento é zerada. Ao encontrar uma pastilha ela é removida do local. Um fantasma não é removido do local quando o Pacman passa sobre sua posição.

Ao encontrar o limite do tabuleiro e o jogador tentar efetuar um movimento ultrapassando este limite, o personagem deve se manter na mesma posição.

Os movimentos são indicados pelas direções norte (N), sul (S), leste (L) e oeste (O).

A figura 1 – a demonstra um tabuleiro com 5 linhas e 5 colunas com dois fantasmas e o Pacman posicionado na posição 1, 1 (linha 1 e coluna 1). A figura 1 – b, demonstra como ficará o tabuleiro após a sequência de movimentos "LLSSSSSSSNOO", com a pontuação acumulada de 4 pontos.

P	*	F	*	*
*	*	*	*	*
*	*	F	*	*
*	*	*	*	*
*	*	*	*	*

Figura 1 – a

		F	*	*
*	*		*	*
*	*	F	*	*
P			*	*
*	*		*	*

Figura 1 – b

Entrada

A entrada contém um caso de teste. A primeira linha do caso de teste contém dois inteiros L e C ($1 \leq L,C \leq 255$) indicando o número de linha e de colunas do tabuleiro. A segunda linha contém um número inteiro N ($0 \leq N \leq L * C$) indicando o número de fantasmas presentes no tabuleiro. As N próximas linhas contém dois inteiros LF ($1 \leq LF \leq L$) e CF ($1 \leq CF \leq C$) indicando a posição de cada um dos fantasmas. A próxima linha contém dois inteiros LP ($1 \leq LP \leq L$) e CP ($1 \leq CP \leq C$) indicando a posição inicial do Pacman que não contem uma pastilha. A última linha do caso de teste contém uma string com até 255 caracteres, indicando os movimentos do Pacman, sendo N para norte, S para sul, L para leste e O para Oeste.

Saída

A saída deverá conter um único número inteiro indicando a pontuação obtida pelo jogador.

Exemplos:

Entradas	**Saídas**
5 5 2 1 3 3 3 1 1 LLSSSSSSSNOO	4

3 3 1 1 3 1 1 LLSOOSLLNN	0

PROBLEMA M

PEGANDO PESOS

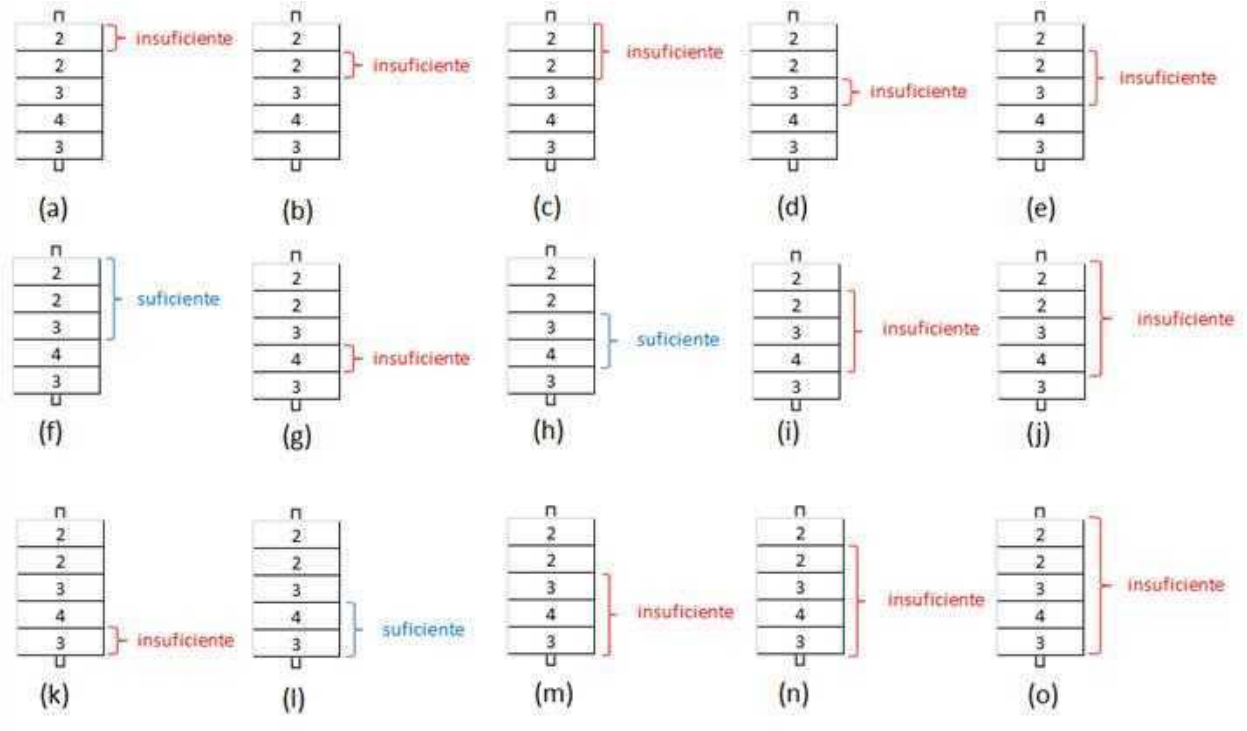
Criado por Jones Mendonça de Souza (IFSP – campus Barretos)
Arquivo: pesos.[c | cpp | cs | java | py]
***Timelimit:** 1*

Marina é treinadora de academia e resolveu criar um método de treinamento que melhore o condicionamento físico dos alunos e estimule o raciocínio matemático. O método visa definir *n* séries de levantamento de pesos fixo.

Observe a ilustração abaixo, Marina passou um treinamento ao seu aluno com 7kg, e disponibilizou a seguinte pilha de pesos (a). Ela deseja saber quantas séries de 7kg o aluno deverá executar. Para isso, o aluno deve seguir estritamente as seguintes regras:

- Desempilhar 1 disco por vez e verificar se consegue realizar uma série com aquele peso;
- É permitido juntar mais de 1 disco para atingir a soma do peso do treinamento, desde que estes já tenham sido desempilhados e a soma tenha sido realizada em ordem consecutiva. Por exemplo, não é permitido somar o primeiro disco do topo com o terceiro, mesmo que a soma seja igual ao do treinamento.

No exemplo abaixo, com um peso de 7kg o aluno conseguirá realizar 3 séries, utilizando os discos de pesos apresentados em (f), (h) e (l).



Seu objetivo é desenvolver um algoritmo que mostre a quantidade de séries que o aluno deverá executar com a quantidade de peso determinada para o treinamento.

Entrada

A primeira linha de entrada contém 2 inteiros separados por um espaço em branco, o primeiro inteiro representa o peso que Marina passou para o treinamento, o outro número inteiro representa a quantidade de discos de peso empilhados na pilha. Depois, seguem-se *k* inteiros representando o peso de cada disco da pilha.

Saída

Seu programa deverá exibir a quantidade de séries que o aluno deverá executar. Caso a pilha não disponha nenhuma configuração para o treinamento, exiba a mensagem “nenhuma”.

Exemplos de Entradas	***Exemplos de Saídas***
7 5 3	3 series

4 3 2 2	
6 5 3 1 3 1 1	1 serie
2 8 1 3 1 3 9 10 1 5	nenhuma

PROBLEMA N
Teclado

Nome do arquivo fonte: teclado.[c | cpp | cs | java | py]
(Adaptado ACM/ICPC por Prof. Ms. Murilo Varges da Silva – IFSP - Birigui)

Quantas teclas são necessárias pressionar para escrever uma mensagem de texto? Você pode pensar que é igual ao número de caracteres no texto, mas isso só é correto se uma tecla gera um caractere. Com dispositivos de bolso, as possibilidades para a digitação de texto são muitas vezes limitadas. Alguns dispositivos fornecem apenas alguns botões, um número significativamente menor do que o número de letras no alfabeto. Para esses dispositivos, o usuário precisa apertar vários botões para digitar um único caractere. Um mecanismo para lidar com essas limitações é um teclado virtual exibido em uma tela, com um cursor que pode ser movido para selecionar caracteres. Quatro botões de seta controlam o movimento do cursor, e quando o cursor é posicionado sobre uma tecla apropriada, pressionando o quinto botão seleciona o carácter correspondente e anexa-o ao fim do texto. Para finalizar o texto, o usuário deve navegar e selecionar a tecla "Enter". Isso fornece aos usuários um conjunto arbitrário de caracteres e que lhes permite digitar texto de qualquer tamanho, com apenas cinco botões de hardware.

Neste problema, você terá um layout de teclado virtual e sua tarefa é determinar o número mínimo de teclas pressionadas para escrever um texto, onde pressionar qualquer um dos cinco botões de hardware conta como uma tecla pressionada. As teclas estão dispostas em uma tabela retangular, de tal modo que cada tecla virtual ocupa uma ou mais unidades da tabela. O cursor começa no canto superior esquerdo do teclado e move-se em quatro pontos cardeais, de tal forma que sempre avança para a próxima unidade quadrada em direção de uma tecla diferente. Se não existe tal quadrado de unidade, o cursor não se move.

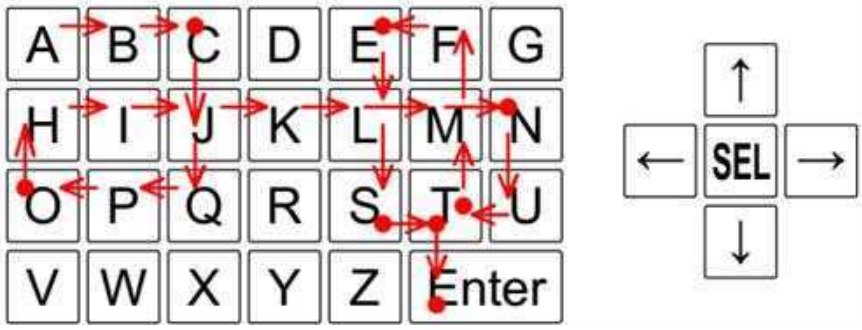


Figura 1 - Exemplo Entrada 1. Um exemplo de um teclado virtual e dos botões em hardware.

Figura 1, ilustra exemplo Entrada 1, mostra uma maneira possível de digitar a palavra "CONTEST" 30 toques em um teclado virtual como o da Figura 1. Os pontos vermelhos representam as teclas virtuais, onde o botão de seleção foi pressionado.

Entrada

A primeira linha da entrada contém dois números inteiros *r* e *c* ($1 \leq r, c \leq 50$), dando o número de linhas e colunas da grade do teclado virtual. O teclado virtual é especificado nas próximas *r* linhas, cada uma das quais contém *c* caracteres. Os possíveis valores desses caracteres são letras maiúsculas, dígitos, um traço e um asterisco (representando Enter). Há apenas uma tecla que corresponde a qualquer caractere dado. Cada tecla é constituída por um ou mais quadrados da tabela que representa o teclado, que irão sempre formar uma região ligada. A última linha da entrada contém o texto a ser digitado. Este texto é uma string não-vazia de, no máximo, 10.000 caracteres, sem asterisco.

Saída

Exibir o número mínimo de toques necessários para digitar o texto todo, incluindo a tecla "Enter" no final. Garantindo que o texto pode ser digitado.

Exemplos:

Entradas	Saídas
4 7 ABCDEFG HIJKLMN OPQRSTU VWXYZ** CONTEST	30
5 20 12233445566778899000 QQWWEERRTTYYUUIIOOPP -AASSDDFFGGHHJJKKLL* --ZZXXCCVVBBNNMM--** -------------------- ACM-ICPC-WORLD-FINALS-2015	160
2 19 ABCDEFGHIJKLMNOPQZY X*****************Y AZAZ	19
6 4 AXYB BBBB KLMB OPQB DEFB GHI* AB	7

Problema O

IFSP Motorsport Championship

Criado por Tiago Dócusse (IFSP – campus Barretos)

Arquivo: motorsport.*[c|cpp|cs|java|py]*

Timelimit: 1s

Hoje em dia existem diversos simuladores virtuais de corridas automobilísticas, e Rafael adora esses simuladores. Recentemente, ele se inscreveu para participar do IFSP Motorsport Championship, um campeonato em que, além de ser um bom piloto, é necessário ser um ótimo engenheiro de motores, já que é possível personalizar alguns aspectos dos motores virtuais antes das corridas. Os motores a combustível de carros podem ser definidos por diversas características, como potência, consumo, torque, entre outros. No caso do IFSP Motorsport Championship, a categoria de carros utilizada no campeonato deste ano é a GT3, que possui carros com diferentes motores e coeficientes aerodinâmicos. Rafael escolheu um carro muito bom aerodinamicamente, porém, com um motor um tanto quanto fraco para essa categoria. Sabendo disso, ele pensou em fazer uma melhoria na curva de torque do seu carro, para ter maior aceleração nas saídas de curva. Rafael estudou a documentação do simulador e descobriu que a curva de torque do carro que ele escolheu foi modelada utilizando um polinômio de segundo grau, que, uma vez atingido o torque máximo, ele decai e é necessário trocar de marcha para que o carro possa ganhar mais torque. Pensando nisso, e sabendo que pode alterar o comportamento do motor do seu carro virtual, Rafael projetou o seu novo motor para que, ao invés de um polinômio de segundo grau, a sua curva de torque seja simulada por um polinômio de quarto grau. Para fazer isso, ele implementou um algoritmo que detecta o torque máximo e aplica um *boost* temporário no motor, na tentativa de aumentar o torque e conseguir mais velocidade com isso, antes que o torque volte a cair novamente. Dessa forma, a curva de torque do motor virtual deve ser comportar, após essa modificação, como um polinômio de quarto grau. Agora que Rafael já definiu o que deseja alterar no motor do seu carro virtual, ele pediu sua ajuda. Ao invés de fazer a alteração e testar o carro, ele quer antes saber se o resultado dará certo, analisando os valores da rotação após a utilização do *boost*, uma vez que tal alteração é complexa e há pouco tempo para o início do campeonato. E aí, você consegue ajudar Rafael a ser o número 1 das pistas virtuais?

Entrada

A entrada do programa deve ser composta inicialmente por uma linha contendo cinco números reais, correspondente aos valores a, b, c, d, e de um polinômio do quarto grau do tipo $ax^4 + bx^3 + cx^2 + dx + e$, tais que $-10^9 \leq a, b, c, d, e \leq 10^9$. Em seguida, informa-se na próxima linha o valor inicial (v_i) e o valor final (v_f) de rotação do intervalo fechado que deseja ser analisado, tais que $-2^{31} \leq v_i, v_f \leq 2^{31} - 1$. Estes valores são números inteiros e podem ser negativos, uma vez que o simulador utiliza valores fictícios para fazer tais cálculos.

Saída

A saída do programa possui duas linhas, a primeira informando o valor (aproximado com sete casas decimais) da rotação no momento em que o torque passa de uma tendência de queda para uma alta, devido à aplicação do *boost*, e a segunda o valor (aproximado com sete casas decimais) da rotação no momento em que o efeito do *boost* acaba e o torque passa de uma tendência de alta para uma tendência de queda. Caso o *boost* não produza a retomada de torque desejada, a saída deve ser formada por uma única linha formada pelas palavras "PARAMETROS FALHOS".

Exemplos de Entradas	***Exemplos de Saídas***
-2 3 1 0 10 -1 1	-0.0982424 0.8482424
1 -5 0 1 -10 -1 1	PARAMETROS FALHOS

Problema P
Expedição Endurance

Criado por: Rafael da Silva Muniz (IFSP - Brangança Paulista)
Arquivo: navio.[c|cpp|cs\java\py]
***Timelimit:** 1*

Fonte: Google Imagens

Em dezembro de 1914 um dos maiores exploradores da Antártica, Ernest Shackleton, iniciava sua segunda expedição ao Polo Sul a bordo de seu navio, o *Endurance*. Porém, no início de 1915 depois de uma forte ventania o *Endurance* foi arrastado ao mar congelante de Weddel. Após ficar mais de 10 meses parado entre dois blocos de gelo, o *Endurance* naufragou. Shackleton tinha que decidir o que fazer o mais rápido possível já que seus suprimentos estavam acabando e ele precisava alimentar diariamente 27 marinheiros e 68 cachorros. A primeira ordem de Shackleton foi de construir uma cozinha e uma cabana com as madeiras que os marinheiros conseguiram resgatar do navio. A segunda ordem foi passada ao cozinheiro para racionar as refeições e o consumo de água. Mesmo estando no meio do gelo, a água doce era restrita e o processo de fusão (transforar gelo em água) consumia muito combustível.

A estratégia criada pelo cozinheiro, para racionar o consumo da água, foi o de lavar a louça somente com o resto da água não consumida na refeição. A ordem da lavagem da louça foi definida pelo cozinheiro como sendo inversa a ordem de entrega na cozinha. Ou seja, a última louça entregue será a primeira a ser lavada, a penúltima entregue será a segunda a ser lavado, a antepenúltima entregue será a terceira a ser lavada. Essa ordem de lavagem deverá ser seguida até a primeira louça entregue que será, respectivamente, a última a ser lavada.

Os marinheiros conseguiram resgatar, antes do naufrágio, 30 itens de cozinha (entre pratos, talheres e canecas) que podem ser usados nas refeições.

Você deve criar um programa para apresentar se todas as louças de uma refeição foram lavadas ou não.

Entrada

A entrada é constituída por **N** linhas. Cada linha pode apresentar os seguintes valores: 1 (indicando que um prato foi entregue ao cozinheiro), 2 (indicando que um talher foi entregue ao cozinheiro), 3 (indicando que um copo foi entregue ao cozinheiro), 0 (indicando que o primeiro item na ordem da lavagem foi finalizado) e -1 (indicando que a quantidade de água disponível acabou).

Saída

Seu programa deve apresentar na saída **-1** caso o cozinheiro tenha lavado toda a louça de uma determinada refeição ou **N** linhas indicando as louças que não foram lavadas naquela refeição.

Exemplos de Entradas	***Exemplos de Saídas***
3 2 2 0 3 1 0 0 1 -1	3 2 1
1 1 2 3 0 0 3 0 2 0 0 2 2 3 1 2 2 0 -1	1 2 2 3 1 2
1 3 0 3 0 0 2 2	-1

1 0 0 0 -1	