10a. EDIÇÃO

MARATONA DE PROGRAMAÇÃO

InterFatecs

Fatec Ribeirão Preto

Centro Paula Souza

SÃO PAULO GOVERNO DO ESTADO

CADERNO DE QUESTÕES

ETAPA ÚNICA

06 de novembro de 2021

APOIO

BANK

POPULOS

MULTILASER

CANAL PRO X

Lago Consultoria

WWW.INTERFATECS.COM.BR

1 Instruções

Este caderno contém 11 problemas – identificados por letras de A até K, com páginas numeradas de 3 até 26. Verifique se seu caderno está completo.

Informações gerais

1. Sobre a competição
   (a) A competição possui duração de 5 horas (início as 09h término as 14h);
   (b) É permitido a consulta a materiais já publicados anteriormente ao dia da competição
   (c) Não é permitido a comunicação com o técnico ou qualquer outra pessoa que não seja a equipe para tirar dúvidas sobre a maratona
   (d) É vedada a comunicação entre as equipes durante a competição, bem como a troca de material de consulta entre elas;
   (e) Cada integrante da equipe poderá utilizar o seu computador/notebook para resolver os problemas, porém, apenas um competidor da equipe ficará responsável pela submissão
   (f) 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 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.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema A

Industria Mágica

Arquivo fonte: industria.{ c | cpp | java | py }
Autor: Lucas Baggio Figueira (FATEC Ribeirão Preto)

Uma determinada indústria dos tempos de outrora deve fazer a separação dos frascos de poção que possuem cores diferentes devido sua finalidade. Um dos gnomos operários dessa indústria deve fazer a separação destes frascos para encaixotá-los, e, como todo o bom gnomo operário ele gosta de diferentes poções para tornar seu trabalho menos entediante, sendo assim ele utiliza uma poção que separa os frascos a cada N sistematicamente até que todos estejam separados. Por exemplo, caso a fila tenha 6 frascos os quais são tirados de 2 em dois até que o frasco 7 seja retirado por último, veja,

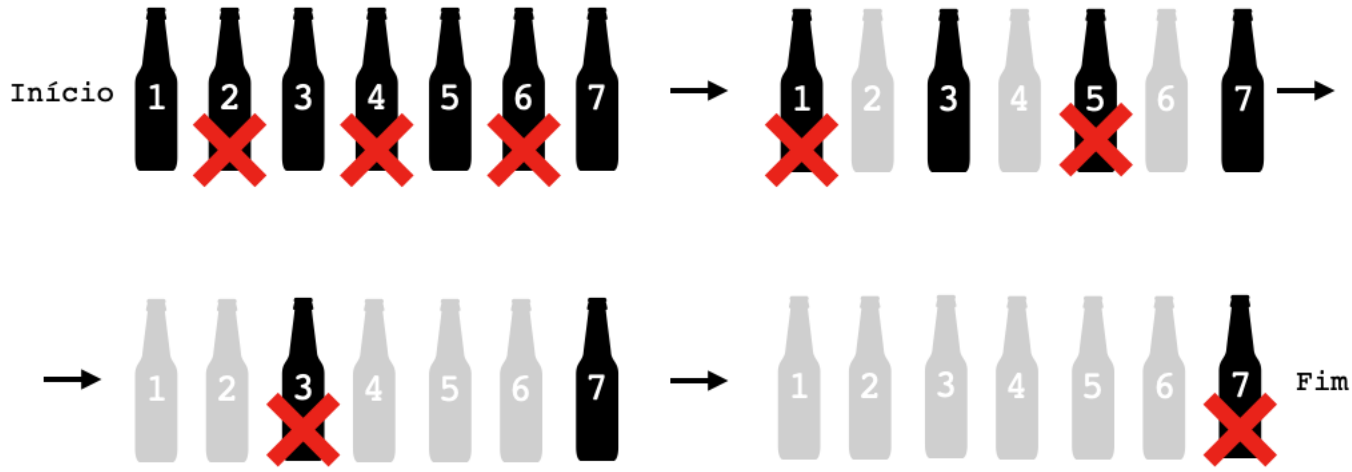


Figura A.1: Sequencia de encaixotamento dos frascos

Entrada

A entrada contém vários casos de teste, cada caso é expresso com 2 inteiros A ($1 < A < 50000$) e B ($1 < B < 5000$), onde A indica a quantidade de garrafas numeradas de forma crescente a partir de 1, e B o tamanho do passo utilizado na remoção sistemática de frascos. A entrada termina com EOF.

Saída

A saída deve, em cada linha, conter o número do último frasco para sua respectiva entrada.

Exemplo de Entrada 1	Exemplo de Saída 1
10 2 6 4 4567 123	5 5 60

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema B

Fluxonator

Arquivo fonte: fluxonator.{ c | cpp | java | py }
Autor: Lucas Baggio Figueira (FATEC Ribeirão Preto)

Um laboratório ultrassecreto tem tentado manipular elétrons de maneira a criar tecnologias, e, portanto, criaram uma armadilha para capturá-los de maneira uniforme, tal armadilha possui três entradas, denominadas A, B, C, as quais recebem elétrons de um meio diverso, e duas saídas denominadas, D e E, que direcionam tais elétrons para um meio controlado. Dentro da armadilha existem 3 alavancas (L_1, L_2, L_3) que direcionam o fluxo de elétrons para alguma das saídas, sendo que cada vez um elétron passa por uma alavanca, a interação eletromagnética faz com que essa alavanca inverta sua posição, e assim, o próximo elétron que passar por ela será direcionado para outro caminho. Na figura abaixo à esquerda é possível ver o fluxo seguido pelo elétron a partir da entrada C, e à direita a armadilha após a mudança das alavancas. Para cada sequência de captura, a armadilha é resetada ficando com L_1, L_2, L_3 virados para a esquerda.

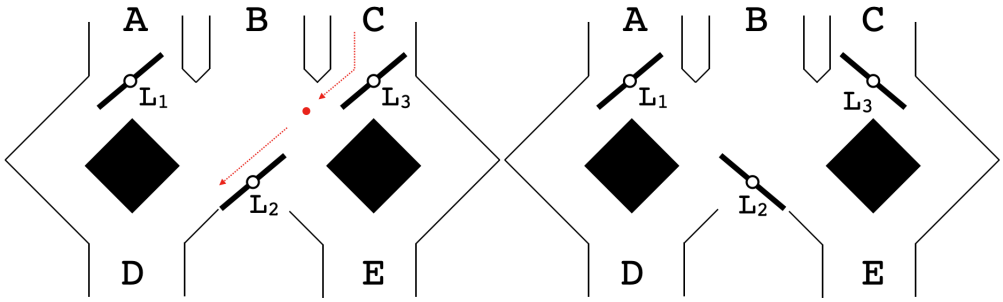


Figura B.1: Fluxonator em ação

Entrada

A entrada contém vários casos de teste, inicialmente tem-se um inteiro N ($1 < N < 1000$) que indica quantos casos de teste serão fornecidos, cada um dos N casos é formado por uma string indicando a sequência de entradas por onde os elétrons são capturados.

Saída

A saída deve, em cada linha, conter uma string indicando a sequência de saída dos elétrons.

Exemplo de Entrada 1	Exemplo de Saída 1
3 ABAA BBBAABCCC BCABCBCAAAAACBBABACCCA	DDED DEDDEDEED DEDDEEDEDDDEEDEDDEDEED

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

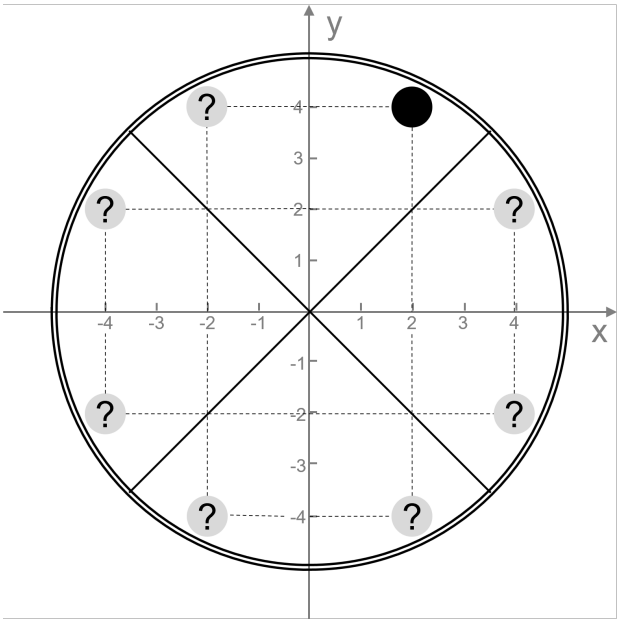
Problema C

Azeitonas na Pizza

Arquivo fonte: azeitonas.{ c | cpp | java | py }
Autor: Rodrigo Plotze (FATEC Ribeirão Preto)

João da Silva é um apaixonado pela Matemática e sempre busca oportunidades de utilizá-la nas mais variadas situações do cotidiano. Um dia desses em sua casa fez o pedido de uma pizza por um aplicativo de delivery, e quando recebeu a pizza ficou pensando no seguinte problema: Existe algum método para determinar simetricamente o posicionamento de oito azeitonas na área do círculo da pizza?

Para ilustrar este problema, observe a imagem:



Na imagem, uma azeitona foi posicionada no *plano cartesiano da pizza* na coordenada $(2, 4)$. Diante disso, como é possível determinar o posicionamento das sete azeitonas restantes de forma a obter uma distribuição simétrica?

Entrada

A entrada começa com uma linha com dois inteiros: N ($1 \leqslant N \leqslant 100$) representando a coordenada (x, y) da posição de uma azeitona no *plano cartesiano da pizza*. A coordenada é válida se, e somente se, $x \neq y$. A última linha de entrada termina com uma quebra de linha.

Saída

Como saída apresente as oito coordenadas, uma coordenada em cada linha, simétricas que representam o posicionamento das azeitonas na pizza. As linhas da saída seguem as posições das azeitonas no sentido horário. Caso a entrada seja inválida, apresente a mensagem *ERRO*. A última linha de saída termina com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
2 4	2 4 4 2 4 -2 2 -4 -2 -4 -4 -2 -4 2 -2 4

Exemplo de Entrada 2	Exemplo de Saída 2
-4 -3	ERRO

Exemplo de Entrada 3	Exemplo de Saída 3
5 1	5 1 1 5 1 -5 5 -1 -5 -1 -1 -5 -1 5 -5 1

Exemplo de Entrada 4	Exemplo de Saída 4
2 9	2 9 9 2 9 -2 2 -9 -2 -9 -9 -2 -9 2 -2 9

Exemplo de Entrada 5	Exemplo de Saída 5
0 2	ERRO

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema D

As Fotos da Ana Maria

Arquivo fonte: fotos.{ c | cpp | java | py }
Autor: Rodrigo Plotze (FATEC Ribeirão Preto)

As fotos da Ana Maria publicadas na rede social não estão ficando com uma qualidade muito boa. Maria tem um estilo fotográfico próprio e adora fotografar paisagens em tons de cinza. Ela já pensou em adquirir um novo smartphone, no entanto, o preço está muito alto e ela não tem condições de realizar a compra neste momento. A Figura apresenta um exemplo de uma fotografia da Maria, em que é possível notar o baixo contraste.

Para auxiliar Ana Maria a entender qual o problema com as suas fotos, você deverá elaborar uma solução capaz de calcular o histograma da imagem. O histograma pode ser definido como um conjunto de números que indicam o percentual de pixels que apresentam um determinado nível de intensidade. Através do histograma é possível analisar a distribuição das intensidades, e assim notar, por exemplo, se a imagem está muito clara, se está muito escura ou com baixo contraste. O resultado poderá ajudar Ana Maria a corrigir problemas de iluminação no momento das fotos. Para calcular o valor de cada elemento do histograma, a seguinte fórmula deve ser utilizada:

$$pr(k) = \frac{n_k}{n}$$

onde: $k = 0, 1, ..., L-1$, onde L é o número de níveis de intensidade da imagem; n = número total de pixels na imagem; $pr(k)$ = probabilidade do k-ésimo nível de intensidade ; n_k = número de pixels cujo nível de intensidade corresponde a k.

Entrada

Um conjunto de valores inteiros,n_k, contendo número de pixels cujo nível de intensidade corresponde a k. Cada valor do conjunto é informado em uma linha separada. A última linha de entrada termina com uma

quebra de linha.

Saída

Como saída apresente um conjunto de valores contendo a probabilidade do *k-ésimo* nível de intensidade. Cada valor resultante deve ser apresentado em uma linha. A última linha de saída termina com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
3156 2284 1365 3678 1576 839 4950 1992 1042 3503 3302 2708	0.104 0.075 0.045 0.121 0.052 0.028 0.163 0.066 0.034 0.115 0.109 0.089

Exemplo de Entrada 2	Exemplo de Saída 2
3937 3199 3315 1927 3901 2757 2668 4239 1378 2063 3575 4856 1409 3610 3906 18 4660 2776 2046 4606	0.065 0.053 0.054 0.032 0.064 0.045 0.044 0.07 0.023 0.034 0.059 0.08 0.023 0.059 0.064 0.0 0.077 0.046 0.034 0.076

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema E

Estacionamento do Seu Zé

Arquivo fonte: estacionamento.{ c | cpp | java | py }
Autor: Rodrigo Plotze (FATEC Ribeirão Preto)

O estacionamento do*Seu Zé* fica localizado no município brasileiro de Passa e Fica, no interior do estado do Rio Grande do Norte. Neste município todas as bicicletas são obrigatoriamente identificadas com placas. Uma placa é definida por um conjunto de 7 (sete) caracteres alfanuméricos, com combinação aleatória de 4 (quatro) letras e 3 (três) números. O estacionamento possui, atualmente, um total de 15 (quinze) vagas identificadas numericamente.

Recentemente o *Seu Zé* implantou no estacionamento um sistema que determina automaticamente a vaga em que a bicicleta deve ser estacionada. O sistema realiza o reconhecimento óptico de caracteres da placa e determina a posição da bicicleta no estacionamento a partir das letras e dos números. Por exemplo, a placa $ABC1D23$ deverá ser estacionada na vaga 12. Para isso, o seguinte cálculo foi realizado:

$$P = 65 + 66 + 67 + 1 + 68 + 2 + 3 = (272\%T) + 1 = 12$$

Em que, os valores 65, 66, 67 e 68 representam os valores decimais dos caracteres A,B,C e D na Tabela ASCII e T o número total de vagas do estacionamento. Quando uma vaga está ocupada não é possível estacionar outra bicicleta, dessa forma, o dono da bicicleta precisará procurar outro estacionamento

Entrada

Uma lista de placas de bicicletas que desejam uma vaga no estacionamento do *Seu Zé*. A última linha de entrada termina com uma quebra de linha.

Saída

A saída deve, em cada linha, apresentar as placas de bicicletas que conseguiram uma vaga no estacionamento. A listagem deve conter o número da vaga e a placa da bicicleta. As demais placas deverão ser ignoradas. A última linha de entrada termina com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
ABC1D23 QNT8B49 JBO5T18 GDK2W13 GXA4D66 RRP4T27 ACP9A44 SLS7B62 GRO2F24 EQY8F35 QGI1Y43 XKN8V47 IJT5M37 TYE7K36 DZH8I89 QIQ3G43 CDO5S18 MUZ2Y29 YEQ4E35 RLG7H29	2 GDK2W13 3 GXA4D66 4 ACP9A44 5 GRO2F24 6 RRP4T27 8 DZH8I89 10 QNT8B49 11 XKN8V47 12 ABC1D23 13 TYE7K36 14 RLG7H29 15 YEQ4E35

MARATONA DE PROGRAMAÇÃO
InterFatecs

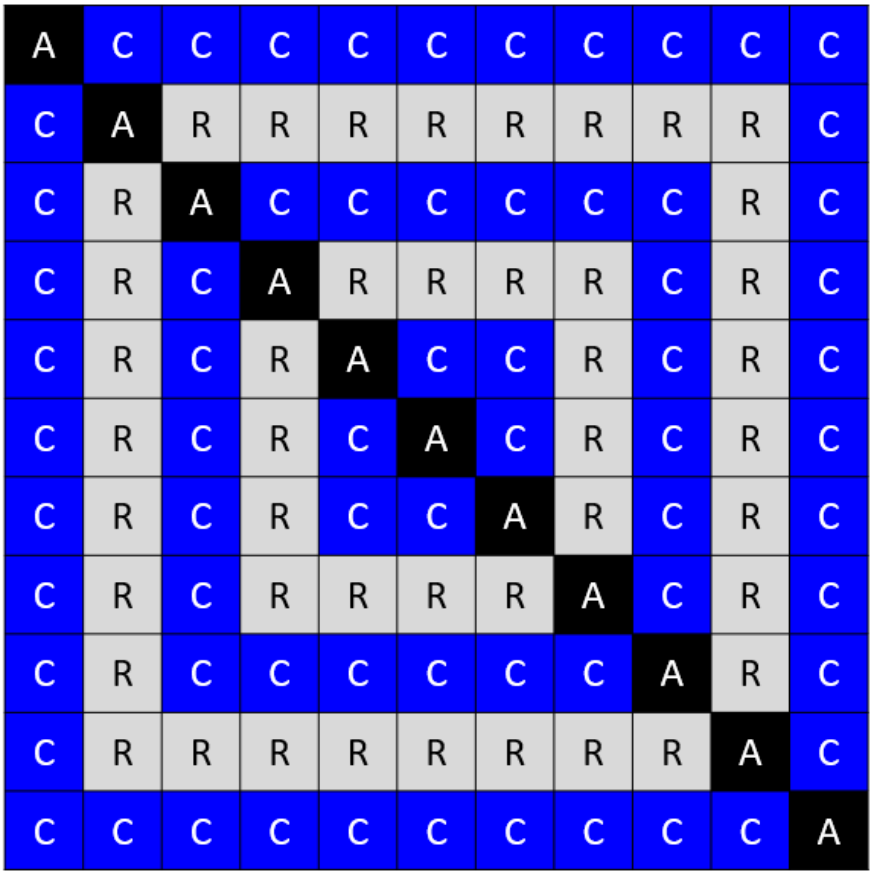
Problema F

SquareCity

Arquivo fonte: squarecity.{ c | cpp | java | py }
Autor: Rodrigo Plotze (FATEC Ribeirão Preto)

Stanley is an architect and is working on the urban design of a new city called *SquareCity*. The city was entirely planned from concentric squares and formed exclusively by houses, streets and avenues. In the project, the houses and streets are arranged concentrically up to the outer limits of the city. A single avenue cuts the main diagonal of the city, and streets separate the blocks of houses.

To help the project visualization, Stanley used graph paper and observed the layout of the houses, identified by the letter *C*, of the streets represented by the letter *R*, and also of the avenue indicated by the letter *A*.



The main difficulty faced by Stanley is to quickly and practically determine the number of houses that can be built in the city. For that, you will have to elaborate a computational solution capable of performing this calculation.

Entrada

The input is composed of a single line containing an odd integer value named x such that $1 \leq x \leq 101$. The value of x represents the number of rows and columns used in the graph paper.

Saída

A single integer value indicating the number of houses that can be built for a city of size x, followed by a line break.

Exemplo de Entrada 1	Exemplo de Saída 1
1	0

Exemplo de Entrada 2	Exemplo de Saída 2
3	6

Exemplo de Entrada 3	Exemplo de Saída 3
9	28

MARATONA DE PROGRAMAÇÃO
InterFatecs

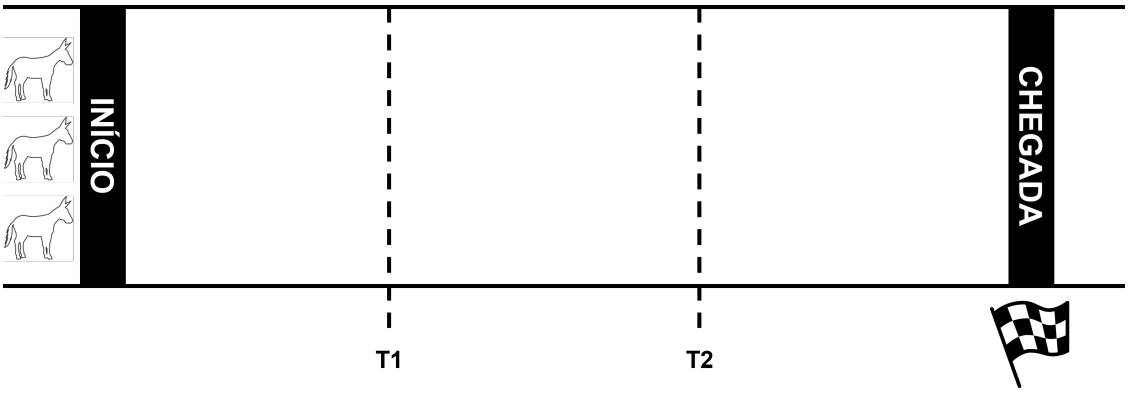
Problema G

De quem é esse Jegue?

Arquivo fonte: jegues.{ c | cpp | java | py }
Autor: Rodrigo Plotze (FATEC Ribeirão Preto)

A Corrida de Jegue é um evento que acontece anualmente na cidade de Itabi, localizada no interior do estado de Sergipe. Este evento atrai milhares de pessoas das mais diversas partes do mundo. Qualquer pessoa pode participar da competição, desde que, é claro, seja capaz de controlar o seu jegue através do percurso de 300 metros rua abaixo.

A cada ano a competição conta com um número maior de inscritos e com isso tem aumentado a dificuldade para determinar os três primeiros colocados da prova. Para resolver este problema a equipe de organizadores pensa em adicionar dispositivos de telemetria nos animais, e assim, realizar a coleta de informações precisas durante a realização da prova. O percurso contará com três pontos de coleta de dados, indicados na como *T1*, *T2* e *CHEGADA*. Em cada ponto de coleta é realizada leitura do tempo de cada participante em milisegundos.



A equipe de organizadores deseja saber quais os três primeiros colocados da competição em cada ponto de coleta. Assim, você deverá escrever uma solução computacional capaz de apresentar os nomes dos três primeiros colocados no T1, T2 e CHEGADA.

Entrada

A entrada é composta por uma lista contendo em cada linha o nome do competidor, o tempo em milisegundos no ponto de coleta T1, o tempo em milisegundos no ponto de coleta T2 e o tempo em milisegundos na linha de chegada. Os dados, em cada linha, são separados por um espaço em branco.

Saída

A saída deve apresentar os nomes dos três primeiros colocados em cada ponto de coleta. Para cada ponto de coleta deve ser apresentado, em uma única linha, o nome do ponto de coleta (T1, T2 ou CHEGADA), o nome do primeiro colocado, o nome do segundo colocado e o nome do terceiro colocado. Por exemplo:

```
T1 João José Maria
T2 Ana João José
CHEGADA João Maria José
```

Exemplo de Entrada 1	Exemplo de Saída 1
Willa 65877 128839 207488 Hayley 65287 124817 193510 Ina 60175 122192 198273 Ezekiel 78636 121501 198047 Desirae 69080 134192 216968 Xandra 62580 136605 198388 Gillian 75148 134331 199639 Pascale 71783 130409 192810 Fay 68518 121211 191793 Roth 65954 131476 214952	T1 Ina Xandra Hayley T2 Fay Ezekiel Ina CHEGADA Fay Pascale Hayley

Exemplo de Entrada 2	Exemplo de Saída 2
Jamal 77261 133536 202025 Nadine 72472 122640 208786 Vance 74614 122907 198825 Shad 72467 127607 217506 Paul 79513 135707 194897 Lyle 72265 122130 191725 Stephen 79377 130456 209896 Benjamin 62816 139920 191728 Paul 65393 129809 188888 Mona 75037 133851 188172 Rashad 74320 121099 205251 Thane 76073 134913 182169 Clio 66988 126774 209863 Kermit 60082 128677 198205 Gareth 65945 133625 191706 Lance 75956 126494 218753 Dorian 67009 128701 204336 Jocelyn 69166 132302 196753 Graham 76864 125035 215506 Raja 66841 133749 181153	T1 Kermit Benjamin Paul T2 Rashad Lyle Nadine CHEGADA Raja Thane Mona

Exemplo de Entrada 3	Exemplo de Saída 3
Colin 64785 135393 194035 Laurel 68326 129286 198369 Linus 79888 131039 205647 Laith 67533 133213 193860 Jin 61684 132355 185737 Zoe 68419 135238 185823 Ella 76993 120402 201008 Castor 72521 125650 219183 Nyssa 63934 138254 186451 Montana 74544 134276 204389 Daryl 62311 136295 184134 Simon 76147 129104 201212 Lewis 66968 138090 210673 Axel 62789 128316 215822 Kenneth 77535 127659 180061	T1 Jin Daryl Axel T2 Ella Castor Kenneth CHEGADA Kenneth Daryl Jin

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema H

AnnaGramas

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

Autor: Sérgio Luiz Banin (FATEC São Paulo e FATEC São Caetano do Sul)

Anna Salas anda aprontando de novo. Aposto que você já ouviu falar dela pois é a mesma garota do problema MegabobageM da Maratona de 2019. Todos sabem que ela adora manipular e combinar caracteres.

Desta vez ela quer voltar a brincar com anagramas e quer sua ajuda com a parte de programação. Um anagrama é uma recombinação de letras para formar uma nova palavra. Por exemplo, PEDRA, PADRE e PERDA são anagramas. É claro que para ser anagrama não é necessário que a recombinação de caracteres produza uma palavra existente no idioma. Desse modo: PEDRA, PADRE, PERDA, ADPRE, ERPDA, ADREP, DERAP e tantas outras possíveis combinações são anagramas obtidos a partir dos caracteres envolvidos.

Dados dois conjuntos de caracteres S1 e S2, o que a Anna deseja agora é saber quantos anagramas de S2 estão contidos em S1 supondo que o tamanho de S1 seja maior ou igual a S2. Veja o exemplo:

S1 = FORORFRDOFR S2 = FOR	**FOR**ORFRDOFR FOR**ORF**RDOFR FORORFRD**OFR**	Existem 3 anagramas de S2 que ocorrem em S1

Figura H.1: Exemplo de Anagramas de S2 que ocorrem em S1

Entrada

A entrada é constituída de vários casos de teste. Na primeira linha haverá um número inteiro NCasos ($1 < NCasos < 50$) que informa a quantidade de pares de cadeias de texto. Em seguida haverá 2*NCasos linhas contendo cadeias de caracteres. A primeira cadeia do par será S1 e a segunda S2. A quantidade de caracteres em cada cadeia varia entre 1 e 10000 ($1 < TamanhoCadeia < 10000$).

Saída

A saída contém um número inteiro ou a palavra "ERRO"(sem as aspas) para cada caso de teste. O número inteiro é a quantidade de ocorrências de anagramas de S2 dentro de S1. A palavra ERRO deve ser exibida quando o tamanho de S2 for maior que o tamanho de S1. Termine a saída com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
6 FORORFRDOFR FOR AABAABAA AABA HOJE NAOEHOJE ABBA BAAB PIRULITOQUEBATEBATEPIRULITOUQUEJABATEU BATE ABCDEFGHIJKLMNOPQRSTUVWXY Z	3 4 ERRO 1 7 0

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema I

Batatinha Frita 1,2,3

Arquivo fonte: batatinha.{ c | cpp | java | py }
Autor: Lucas Baggio Figueira (FATEC Ribeirão Preto)

Um grupo aficionado por jogos extremos criou um desafio de vida ou morte, onde o competidor tem, no momento que ouvir Batatinha Frita 1,2,3, que encontrar o caminho mais curto por meio de um complicado labirinto. Assim que o competidor em questão chegar à saída do labirinto ele deverá acionar uma alavanca que impede que uma quantidade potencialmente letal de gás mostarda seja despejada no ambiente em questão. Portanto, os competidores que tenham a coragem de se submeter à este desafio devem usar toda a sua intuição, agilidade e capacidade física.

Entrada

A entrada contém vários casos de teste, configurada da seguinte maneira, na primeira linha tem-se um inteiro N (1 < N < 100) que indica o tamanho do labirinto (N x N), na linha subsequente tem-se o ponto de entrada formado por a (0 < a <= N) e b (0 < b <= N) representando, respectivamente, linha e coluna onde o competidor deverá entrar no labirinto, logo abaixo tem-se c (0 < c <= N) e d (0 < d <= N), representando a linha e coluna, respectivamente, de onde há a alavanca de segurança. Por fim, tem-se o labirinto (N x N) onde cada posição pode conter 0 ou 1, sendo que 1 indica uma passagem válida no labirinto.

Entrada

A saída deve conter a sequência do caminho mais curto entre (a,b) e (c,d).

Exemplo de Entrada 1	Exemplo de Saída 1
10 10 6 2 10 0000000000 1100011111 0110010000 0011110000 0010011110 1110000010 0011110000 0000011111 0000010000 0000010000	10 6 9 6 8 6 7 6 7 5 7 4 7 3 6 3 5 3 4 3 4 4 4 5 4 6 3 6 2 6 2 7 2 8 2 9 2 10

Exemplo de Entrada 2	Exemplo de Saída 2
5 1 3 5 5 00100 00111 10101 11101 00001	1 3 2 3 2 4 2 5 3 5 4 5 5 5

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema J

Power Up Battle (Season #1)

Arquivo fonte: power.{ c | cpp | java | py }
Autor: Fabrício Gustavo Henrique (FATEC Ribeirão Preto)

Power Up Battle é um jogo de cartas que envolve várias magias para aumentar a força do seu exército. Claro, o objetivo é reduzir os pontos de vida do seu oponente a zero. Para medir e treinar o nível dos jogadores surgiu a ideia de desenvolver um "Companion App" que ajuda a identificar qual a maior quantidade de dano possível de ser aplicada em uma jogada. Caso esse jogador atinja essa quantidade máxima de forma constante, certamente isso faz dele um jogador mais experiente.

Essa quantidade ótima de dano é calculada com base na quantidade de cavaleiros aliados no campo de batalha, cada um deles com um poder de ataque inicial de 1 (um), e as cartas/magias que o jogador possui em mãos. Assim, o desafio é identificar qual será a melhor combinação de jogada (ordem em que as cartas/magias são lançadas) de modo a maximizar a quantidade de dano causado pelos cavaleiros?

Nesta temporada, o jogo possui as seguintes cartas:

- 'T' (Treinamento). Cada cavaleiro aliado no campo de batalha recebe +1 de ataque.
- 'R' (Reunir). Cada cavaleiro aliado no campo de batalha recebe +N/2 de ataque, sendo N a quantidade de aliados em campo.
- 'S' (Sacrificar). Sacrifique um cavaleiro. Cada cavaleiro aliado restante recebe +A/2, sendo A o poder de ataque da criatura sacrificada.

A quantidade de cavaleiros no campo de batalha é definida com três dados D20 e a quantidade de cartas compradas do grimório de cartas é definida por um D8.

Dessa forma, deve ser informado para o "Companion App" a quantidade 1 <= N <= 60 de cavaleiros aliados no campo de batalha e a quantidade 1 <= C <= 8 de cartas em mãos. Em seguida, a lista de cada uma das cartas. Como resultado, o "Companion App" deve apresentar o máximo poder de ataque, ou seja, a soma do poder de ataque de cada cavaleiro depois de aplicar as cartas/magias em mãos. Lembrando que não é obrigatório utilizar todas as cartas que estão em mãos, uma vez que nem sempre isso é garantia de maximizar o ataque.

Abaixo segue um exemplo dos dados informados para o "Companion App":

4 3 R T T

Com base nos dados acima temos quatro cavaleiros, todos com o poder inicial de ataque 1 (um), ou seja, termos no campo de batalha "1 1 1 1". Após lançar a primeira carta, 'R', cada cavaleiro recebe +4/2 de poder, resultando no seguinte estado do campo de batalha "3 3 3 3". Ao aplicar a próxima carta 'T', teremos "4 4 4 4" e em seguida com outra carta 'T', "5 5 5 5". Somando o poder de todos os cavaleiros teremos um dano máximo de 20, resultado que deve ser apresentado pelo "Companion App". Vale lembrar que as cartas devem ser utilizadas na melhor ordem possível, não necessariamente na apresentada na entrada.

Exemplo de Entrada 1	**Exemplo de Saída 1**
4 3 R T T	20

Exemplo de Entrada 2	Exemplo de Saída 2
5 3 R S T	24

Exemplo de Entrada 3	Exemplo de Saída 3
5 2 S S	5

Exemplo de Entrada 4	Exemplo de Saída 4
21 5 S T S R S	720

Exemplo de Entrada 5	Exemplo de Saída 5
33 5 T S T T R	960

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema K

Programando o Resgate

Arquivo fonte: resgate.{ c | cpp | java | py }
Autor: Fabrício Gustavo Henrique (FATEC Ribeirão Preto)

"A tarefa é simples: temos que sair sem ser visto!" (TeamBrabo)

Uma equipe de resgate precisa de apoio para planejar o salvamento de reféns. É preciso saber se um resgate sem enfrentamento é possível. Para isso, a equipe busca ajuda técnica para o desenvolvimento de um algoritmo capaz de responder essa pergunta rapidamente.

A ideia é que a partir de um número N de cômodos de um determinado local, seja possível dizer se é ou não possível traçar uma rota do cômodo R, onde os reféns estão localizados, até o cômodo S, que dá acesso à saída. Claro, que tudo isso deve ser feito evitando os cômodos onde existem terroristas. Com base nesses dados e na especificação de acesso (portas) entre os cômodos, o TeamBrabo pede apoio para o resgate dos próximos inocentes.

Entrada

A primeira linha da entrada contém a quantidade $3 \leq N \leq 50$ de cômodos, o número $1 \leq R \leq N$ que identifica a localização dos reféns e o número $1 \leq S \leq N$ que identifica a localização do cômodo de saída. Em seguida segue o valor de $1 \leq T \leq N$ que especifica quantidade de cômodos com terroristas, sendo listados na próxima linha os T números representando cada um dos cômodos com a presença de terroristas. Por fim, segue a especificação da quantidade de $1 \leq P \leq N\frac{(N-1)}{2}$ de portas, ou seja, dos acesso entre os cômodos, seguido de P linhas contendo um par de números representando a existência de uma passagem entre os cômodos.

Saída

Na saída deve ser impresso uma linha contendo "ABORTAR", caso não seja possível um resgate sem enfrentamento ou "PROSSEGUIR", caso seja possível ir até o cômodo de saída sem ter contato com os terroristas.

Exemplo de Entrada 1	Exemplo de Saída 1
5 5 4 1 2 3 5 3 3 2 1 4	ABORTAR

Exemplo de Entrada 2	Exemplo de Saída 2
6 1 6 2 2 3 7 1 2 1 4 2 3 2 5 3 6 4 5 5 6	PROSSEGUIR