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

IV MARATONA DE PROGRAMAÇÃO INTERIF 2021

ETAPA ÚNICA

06/11/2021 – On-line

Caderno de Problemas

Informações Gerais

Este caderno contém treze problemas, as páginas estão numeradas de 1 a 17, não contando esta página de rosto. Verifique se o caderno está completo.

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 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)
- C++ - gcc version 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)
- Python 3 - Python 3.8.10
- Java - openjdk-11.0.11
- C# - mono JIT 6.12

Problema A
Esteganografia

Por Cássio Agnaldo Onodera (IFSP – campus Birigui)

Arquivo: trade.[c|cpp|java\cs\py]

***Timelimit:** 1*

Pouco tempo após o surgimento da escrita se viu a necessidade de disfarçar mensagens de forma que somente o destinatário pudesse compreender. Durante as guerras que ocorreram na antiguidade, os generais precisavam dar ordens a seus soldados sem que essas pudessem ser interceptadas e compreendidas por seus inimigos. Este é apenas um exemplo, mas a necessidade de disfarçar mensagens pode ser encontrada em diversas áreas, tal como um político trocar mensagens com seus aliados, um namorado enviando cartas para sua namorada etc. Duas técnicas foram utilizadas para disfarçar mensagens: a criptografia e a esteganografia. A criptografia altera a mensagem original e mesmo que o opositor tenha acesso a ela, não conseguiria entendê-la de imediato. Já na esteganografia a mensagem se mantém a mesma, e são utilizadas técnicas para escondê-la. Heródoto conta a história de um grego que precisava transmitir uma mensagem secretamente, então ele raspa o cabelo do mensageiro, tatua a mensagem na cabeça raspada e espera que o cabelo cresça novamente. Ao chegar ao destinatário, o mensageiro raspa a cabeça, revelando a mensagem. Obviamente que esta técnica é totalmente inviável. Com o advento dos computadores e algoritmos modernos várias técnicas foram criadas, que permitiram esconder textos em arquivos com outra finalidade, tal como esconder textos em arquivos de imagens, áudio, vídeo etc.

Em 1605, Francis Bacon criou o método de esteganografia conhecido como Código de Bacon, em que utiliza um texto falso para injetar uma mensagem. Nesta técnica, cada letra da mensagem é substituída por um conjunto de 5 caracteres binários que pode ser "A" e "B".

A = AAAAA	H = AABBB	O = ABBBA	V = BABAB
B = AAAAB	I = ABAAA	P = ABBBB	W = BABBA
C = AAABA	J = ABAAB	Q = BAAAA	X = BABBB
D = AAABB	K = ABABA	R = BAAAB	Y = BBAAA
E = AABAA	L = ABABB	S = BAABA	Z = BBAAB
F = AABAB	M = ABBAA	T = BAABB	
G = AABBA	N = ABBAB	U = BABAA	

Tabela – Código de Bacon

Desta forma, substituímos o "A" da mensagem por "AAAAA", um "B" por "AAAAB" e assim sucessivamente. Por exemplo, se desejamos esconder a palavra "IFSP", ficaria assim:

Mensagem original: I F S P

Mensagem cifrada: ABAAA AABAB BAABA ABBBB

Desta forma, podemos esconder a mensagem em um texto falso substituindo a letra "A" por uma letra minúscula e a letra "B" por uma letra maiúscula. Por exemplo:

Mensagem cifrada: ABAAA AABAB BAABA ABBBB

Texto falso: REDE FEDERAL DE EDUCACAO

Texto final: rEde fedErAL de EduCACAo

O destinatário que receber a mensagem deve transformar cada letra minúscula na letra “A” e cada letra maiúscula na letra “B” e separar em grupos de 5 letras. Utilizando a tabela do Código de Bacon, transformar cada grupo de letras com “A” e “B” em uma letra e terá a mensagem original.

Este método foi criado 1605 quando não existia computadores e as conversões eram feitas de forma manual. Além disto, utilizando blocos binários com 5 caracteres, só seria possível representar 32 símbolos diferentes, o que seria impossível de representar as letras maiúsculas e minúsculas, os números, os caracteres especiais etc. Atualmente os computadores já possuem um código para cada letra do alfabeto (maiúsculas e minúsculas), números e outros caracteres (conhecido como Tabela ASCII) e para representá-los são utilizados 8 bits que permitem a representação de 256 símbolos diferentes. Por exemplo:

Símbolo	Decimal	Binário
A	65	01000001
B	66	01000010
Z	90	01011010
a	97	01100001
b	98	01100010
z	122	01111010
0	48	00110000
8	56	00111000
?	63	00111111
*	42	00101010

Utilizando o código ASCII, em binário, de cada caractere, podemos adaptar o Código de Bacon para inserir mensagens secretas em textos falsos.

Bob, recebeu uma mensagem de Alice escondida dentro de um texto falso inserido através desta adaptação do Código de Bacon e está com dificuldade para obtê-la. Faça um programa que receba um texto e obtenha a mensagem secreta.

Entrada

A única entrada do programa é um texto com até 2500 caracteres.

Saída

A saída é a mensagem secreta.

Exemplos de Entradas	Exemplos de Saídas
a MaraToNa INtERIf e UM eveNTo DO iNSTItUTo FEDerAL De EducACAo, cIenCIa E TECnoLogia de SAo pauLo QUe FOi BaSEada nA mARaTONa dE PrOgramACao da SocIedade bRAsILEirA De comPuTACao. eStE EVeNtO e UMa cOmPeTIcAO de pROgrAmAcaO de compUTaDOres DEsTINAdA Aos aLUnOS dos cUrSOs DA area de informatica dos campi do instituto federal de sao paulo (tecnicos e superiores).	Encontro amanha naquele local

iNstITUTo FedERal dE educaCao	OLA

eSTEgANogRAfIA E O eSTudo E UsO Das TeCniCas para OCUlTAr a EXisteNcIA dE umA meNSAGEM deNtro de oUtRa, umA fORMa De SeGUraNcA poR obscurANtIsmo. o PRiMEIRo USO reGisTRado dA pALAvrA DatA DO ANO de 1499, no livro steganographia, de johannes trithemius.	voce vai? Que horas?

Problema B
Otto e seus palíndromos

Por Jorge Francisco Cutigi (IFSP – campus São Carlos)

Arquivo: palindromos.[c|cpp|java\cs\py]

***Timelimit:** 1*

Otto gosta muito de estudar. Na aula de português ele aprendeu o conceito de palíndromo. Ele ficou muito feliz, já que percebeu que seu nome também se tratava de um palíndromo. A professora explicou que uma palavra é um palíndromo se ela pode ser invertida e não modificar a palavra original. Por exemplo, a palavra ARARA é um palíndromo.

Agora Otto vive procurando palavras palíndromas. Ele acha que já conseguiu listar todas as existentes na língua portuguesa. Não satisfeito, agora Otto tem um novo passatempo. A cada palavra, ele verifica se rearranjando os caracteres, ela poderia ser um palíndromo. Por exemplo, a palavra MORARAM pode ter os caracteres rearranjados para formar a palavra MARORAM, que é um palíndromo.

Otto está aprendendo a programar e como exercício decidiu criar um programa que, dado uma palavra, informa se ela pode ser rearranjada para formar um palíndromo ou não.

Entrada

A entrada consiste de apenas uma linha que contém uma string S. Assuma que todas as letras de S não terão acentos e serão todas maiúsculas ou todas minúsculas.

Saída

Seu programa deve imprimir o número inteiro 1 caso a string S possa ser rearranjada para formar um palíndromo e 0 caso contrário.

Restrições

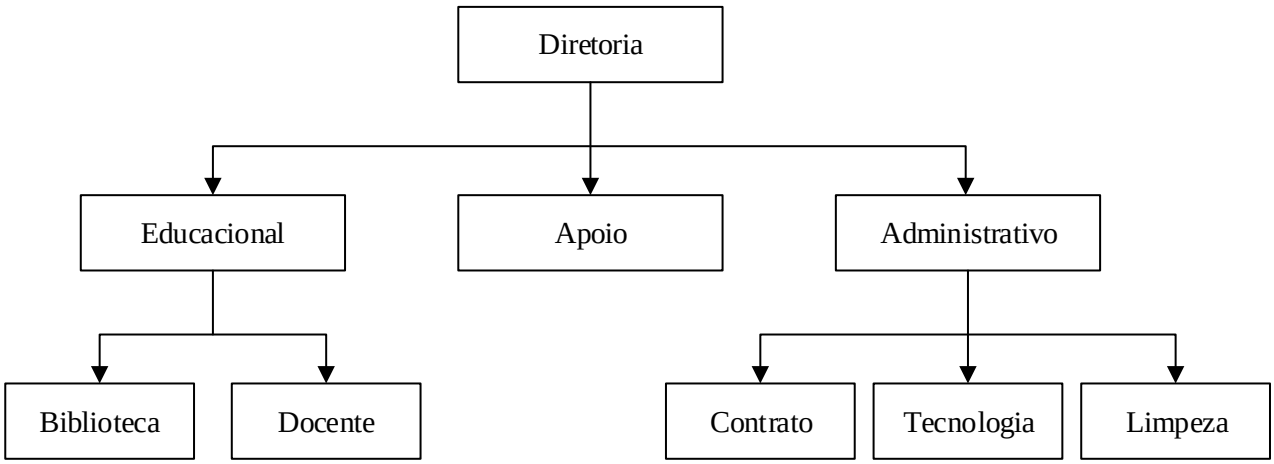
$1 \leq |S| \leq 50$.

Exemplos de Entradas	Exemplos de Saídas
moraram	1
algoritmo	0

Problema C
Organograma

Por Tiago Alexandre Dócusse (IFSP – campus Barretos)
Arquivo: organograma.[c|cpp|java\cs\py]
***Timelimit:** 1*

A escola em que você estudou durante toda a sua infância precisa fazer um relatório dos seus setores. No entanto, eles não possuem esses dados digitalizados, e precisam de um programa que faça isso de maneira rápida e prática. Como você é um excelente programador e tem um carinho especial pela sua escola, resolveu ajudá-los nesta questão. A escola possui setores organizados de maneira hierárquica, sendo representados como um organograma, onde cada setor é responsável por zero ou mais setores, que podem ter setores sob sua responsabilidade. Abaixo é exibido um exemplo de um organograma, onde a seta indica responsabilidade de um setor para o outro.



Entrada

A entrada do programa deve ser o nome do setor mais alto na hierarquia. Em seguida, seguem-se várias linhas no formato “nome1 nome2”, sendo que o setor denominado nome2 é o setor responsável pelo setor nome1. O término da entrada de setores é realizado ao se informar a cadeia de caracteres “fim entrada”. Por fim, deve-se ler o nome de um setor a ser pesquisado. Todos os nomes possuem menos de 40 caracteres, não existem setores com nomes repetidos e não existem setores com nomes contendo mais de uma palavra.

Saída

Deve-se exibir o setor informado juntamente com o nome de todos os setores sob a responsabilidade do setor informado na entrada, em ordem alfabética. Caso um setor sob a responsabilidade do setor informado possua setores sob sua responsabilidade, deve-se exibir esses setores em ordem alfabética antes de exibir os próximos setores. Cada setor deve ser exibido em uma linha.

Exemplos de Entradas	Exemplos de Saídas
Diretoria Educacional Diretoria Apoio Diretoria Administrativo Diretoria Biblioteca Educacional Contrato Administrativo Tecnologia Administrativo Docente Educacional Limpeza Administrativo	Educacional Biblioteca Docente

fim entrada Educacional	

Diretoria Educacional Diretoria Apoio Diretoria Administrativo Diretoria Biblioteca Educacional Contrato Administrativo Tecnologia Administrativo Docente Educacional Limpeza Administrativo fim entrada Diretoria	Diretoria Administrativo Contrato Limpeza Tecnologia Apoio Educacional Biblioteca Docente

Problema D
Problema de Syracuse

Por Claudio Haruo Yamamoto (IFSP – campus Salto)
Arquivo: syracuse.[c|cpp|java\cs\py]
***Timelimit:** 1*

Existe um problema chamado Problema de Syracuse, que consiste em aplicar duas regras a um número inteiro. Primeira regra: se o número for par, então divida-o por 2. Segunda regra: se o número for ímpar, então multiplique-o por 3 e some 1. Seguindo esta regra sucessivamente, sempre se chega ao número 1 (pelo menos para os números menores que $19 * 2^{58}$). Tomando o número 11, por exemplo, aplica-se a primeira regra: 11 * 3 + 1 = 34. 34 é par, então 34 / 2 = 17. Por sua vez, 17 é ímpar, então 17 * 3 + 1 = 52. E desta maneira seguem 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. A partir daí, começa o ciclo 4, 2, 1, 4, 2, 1, 4, 2, 1.

Dado um número N, determine a quantidade de passos, aplicando-se as regras do Problema de Syracuse, até chegar ao número 1 pela primeira vez.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste contém um número inteiro N ($1 <= N <= 10^6$). A entrada termina com EOF.

Saída

A saída consiste em uma linha para cada caso de teste contendo a quantidade de passos, aplicando-se as regras do Problema de Syracuse, até chegar ao número 1 pela primeira vez.

Exemplos de Entradas	Exemplos de Saídas
1 2 3	0 1 7

Problema E
Matemática

Por Tiago Alexandre Dócusse (IFSP – campus Barretos)
Arquivo: matematica.[c|cpp|java\cs\py]
***Timelimit:** 1*

Rafael é um brilhante aluno e adora matemática. Um dia, fazendo um treinamento para a olimpíada de matemática, se deparou com o seguinte problema: Uma função *f* é definida da seguinte forma:

$$f(x) = \begin{cases} x - 3, & x \geq 10.000.000 \\ f\left(f\left(f(x+13)\right)\right), & x < 10.000.000 \end{cases}$$

Rafael até conseguiu calcular o resultado da função para alguns valores de *x*. No entanto, ele não possui todas as respostas, e pediu a sua ajuda para criar um programa que calcule as respostas da função. Você consegue ajudar Rafael no seu treinamento?

Entrada

A entrada é formada por um único número inteiro *x* que deve ser usado para calcular o resultado da função, sendo $0 < x < 20.000.000$.

Saída

A saída exibe o resultado da função para o número de entrada fornecido.

Exemplos de Entradas	Exemplos de Saídas
15623145	15623142
10000000	9999997

Problema F
Combustível

Por Felipe Gobo Bruno (IFSP – campus Boituva)
Arquivo: combustivel.[c|cpp|java\cs\py]
***Timelimit:** 1*

O IFSP Campus Boituva está buscando economizar o máximo possível no abastecimento dos veículos oficiais.

Buscando uma solução para esse problema, um servidor do campus encontrou uma matéria ensinando uma maneira para descobrir qual combustível (etanol ou gasolina) é mais rentável financeiramente, mas ele não possui conhecimento computacional para tornar essa análise eficaz.

O servidor relatou que para o etanol ser mais vantajoso ele deve custar menos que 70% do valor da gasolina, ou seja, dividindo o valor do etanol pelo valor da gasolina o quociente deverá ser menor que 0,7.

De acordo com as informações fornecidas acima, ajude o IFSP Campus Boituva por meio do desenvolvimento de uma solução computacional para descobrir qual combustível (etanol ou gasolina) é mais rentável financeiramente.

Entrada

A primeira linha da entrada possui um número ponto flutuante (reais) com precisão simples V ($0 \leq V \leq 10^{38}$), representando o valor da gasolina.

A segunda linha da entrada possui um número ponto flutuante (reais) com precisão simples T ($0 \leq T \leq 10^{38}$), representando o valor do etanol.

Saída

A saída deve exibir uma única linha contendo GASOLINA para os casos em que a gasolina seja mais vantajosa ou ETANOL para os casos em que o etanol seja mais vantajoso.

Exemplos de Entradas	Exemplos de Saídas
5.92 4.64	GASOLINA
6.50 4.25	ETANOL
7.00 4.43	ETANOL
5.98 5.01	GASOLINA
6.29 5.29	GASOLINA

Problema G
Super múltiplos de 3

Por Márcio Kassouf Crocomo (IFSP – campus Piracicaba)
Arquivo: multiplos.[c|cpp|java\cs\py]
***Timelimit:** 1*

Luigi é um entusiasta da matemática que um dia descobriu uma coisa que achou fantástica: uma forma de verificar se um número é divisível por três a partir da soma de seus algarismos. A regra é que um número é divisível por 3 se, e somente se, a soma de seus algarismos é um número múltiplo de 3. Luigi percebeu que, a partir dessa regra, era possível verificar se números muito grandes eram divisíveis por 3. Não contente com o novo conhecimento adquirido, Luigi começou a tentar compreender a relação entre os números e chamou seu irmão, Mario, para ajudá-lo com uma atividade. Dado um número, seu irmão Mario realizava a soma de todos os algarismos pares desse número e verificava se esta soma era um múltiplo de três. Por sua vez, Luigi repetia a verificação, mas levando em conta a soma apenas dos algarismos ímpares. Como pretendiam repetir o experimento para uma grande quantidade de números, os irmãos pediram sua ajuda para criar um programa que automatize a tarefa explicada.

Entrada

Um número inteiro representado por no mínimo 1 e no máximo 20 dígitos decimais.

Saída

Duas linhas, sendo que na primeira linha deve estar escrito a letra S caso a soma dos dígitos pares do número em questão seja um valor divisível por 3, ou a letra N caso contrário. Na segunda linha a letra S deve aparecer caso a soma dos dígitos ímpares do número em questão seja um valor divisível por 3, ou a letra N caso contrário.

Exemplos de Entradas	Exemplos de Saídas
123	N N
2223	S S
1118	N S
1	S N
6	S S
7	S N
33333333333333333333	S S

Problema H
Vacinômetro

Por Felipe Gobo Bruno (IFSP – campus Boituva)
Arquivo: vacinometro.[c|cpp|java\cs\py]
***Timelimit:** 1*

A população da cidade de Boituva solicitou à secretaria de saúde do município que fosse criado um mecanismo para disponibilizar a porcentagem de habitantes vacinados na cidade.

Para atender a demanda da população boituvense foi idealizado o Vacinômetro, uma ferramenta que exibe a porcentagem de habitantes vacinados na cidade.

As secretarias de saúde das cidades vizinhas ficaram sabendo dessa ferramenta, agora a secretaria de saúde de Boituva precisa da sua ajuda para desenvolver essa solução computacional para disponibilizar a todas as cidades interessadas em exibir essa informação para sua população.

Entrada

A primeira linha da entrada possui um número inteiro N ($1 \leq N \leq 2.147.483.647$), representando a quantidade de habitantes da cidade.

A segunda linha da entrada possui um número inteiro V ($1 \leq V \leq 2.147.483.647$), representando a quantidade de habitantes vacinados na cidade.

Saída

A saída deve exibir uma única linha contendo um número ponto flutuante com precisão de 1 (uma) casa decimal P ($0 \leq P \leq 10^{38}$), representando a porcentagem de vacinados na cidade, seguido do símbolo %.

Exemplos de Entradas	Exemplos de Saídas
64751 38452	59.4%
1248977 847996	67.9%
886339 774123	87.3%
9645 8553	88.7%
246357 66821	27.1%

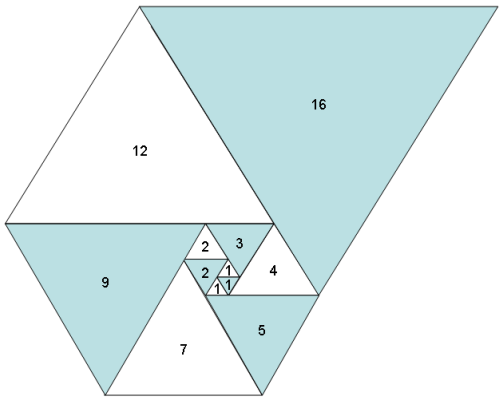
Problema I
N-ésimo termo

Por Claudio Haruo Yamamoto (IFSP – campus Salto)

Arquivo: termo.[c|cpp|java\cs\py]

***Timelimit:** 1*

Em uma espiral de triângulos, cada triângulo compartilha um lado com dois outros, conforme apresentado na figura.



Assim, é formada a sequência 1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, ...

Dado um número N, encontre o N-ésimo termo desta sequência.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste contém um número inteiro N ($1 \leq N \leq 159$). A entrada termina com EOF.

Saída

A saída consiste em uma linha para cada caso de teste contendo o N-ésimo termo da sequência apresentada.

Exemplos de Entradas	Exemplos de Saídas
4 7 5 10 1 22 15 50 80	2 4 2 9 1 265 37 696081 3208946545

Problema J
Pega Varetas

Por Edmar Santos (IFSP – campus Birigui)
Arquivo: varetas.[c|cpp|java\cs\py]
***Timelimit:** 1*

Em um dia chuvoso, sem saber o que fazer, três sobrinhos de Miguel resolveram brincar de "pega varetas" na grande varanda da casa em frente ao jardim. No jogo de "pega varetas" o objetivo é retirar as varetas do monte sem mexer as demais, a fim de somar o maior número de pontos do que os oponentes. Depois que as varetas são espalhadas em um monte, o jogador que inicia o jogo remove quantas varetas puder evitando mexer nas demais. Assim que o jogador mexe numa vareta que não seja a que está retirando, perde a vez, e o próximo jogador inicia sua tentativa de retirada. O jogo termina quando não houver mais varetas e ganha quem somar mais pontos. Como nenhum deles possuía um jogo de varetas completo, eles decidiram juntar o que cada um tinha nas suas latinhas. Também decidiram que cada vareta amarela valeria 5 pontos, as verdes 10, azuis 15, vermelhas 20 e as pretas 50 pontos. Ao ver os garotos brincando, Miguel, que já cursou disciplinas de programação, resolveu fazer um programa para contabilizar as jogadas dos meninos para apresentar o ganhador e sua pontuação, ou mesmo se houve empate. Ajude Miguel a desenvolver esse programa.

Entrada

A entrada possui vários casos de testes. Cada caso de teste é composto por duas linhas. Na primeira linha do caso de teste há somente um número inteiro N ($1 \leq N \leq 3$), que indica qual jogador iniciará a rodada ou N = (-1) o que significa que não há mais rodadas para contabilizar e o programa deverá ser encerrado. A segunda linha contém vários números inteiros V ($-1 \leq V \leq 5$) separados por um espaço onde: V = 1 representa a retirada de uma vareta amarela pelo jogador atual, V = 2 uma vareta verde, V = 3 uma vareta azul, V = 4 uma vareta vermelha, V = 5 uma vareta preta. Quando V = 0 indica que o jogador atual mexeu o monte de varetas perdendo a vez, então a jogada seguinte deverá ser do próximo jogador. O valor V = (-1) representa que não há mais varetas para ser retirado e é o fim da rodada atual.

Saída

A saída consiste em três linhas para cada caso de teste. A primeira linha mostra o número da rodada. A segunda linha mostra a pontuação do ganhador (ou dos ganhadores se houver empate). Já a terceira linha mostra o(s) ganhador(es). Todas as linhas devem ser formatadas conforme o exemplo de saída. Deve ser deixado uma linha em branco somente entre os casos de teste.

Exemplos de Entradas	Exemplos de Saídas
2 0 4 0 0 1 -1 3 2 5 0 4 0 2 3 5 0 1 1 1 -1 1 3 2 2 4 0 1 4 3 1 0 3 2 0 1 4 -1 1 2 2 1 4 1 1 3 4 3 3 1 1 4 5 5 0 5 3 -1 -1	RODADA 1 Ganhador com 20 pontos Jogador 3 RODADA 2 Empate com 75 pontos Jogador 2, Jogador 3 RODADA 3 Ganhador com 80 pontos Jogador 1 RODADA 4 Ganhador com 250 pontos Jogador 1

Problema K
Isabel, o bolo e o menor preço

Por Jorge Francisco Cutigi (IFSP – campus São Carlos)
Arquivo: bolo.[c|cpp|java\cs\py]
***Timelimit:** 1*

Na cidade de Ifspolândia há uma padaria muito famosa por seus bolos, que são conhecidos nacionalmente. Nessa padaria, os bolos são de determinados tipos e são produzidos e cortados em pedaços iguais. Acontece que, estranhamente, para cada bolo a padaria cobra do cliente valores (em R$) diferentes, dependendo da quantidade de pedaços que o cliente compra. Por exemplo, a tabela a seguir representa a tabela de valores para um bolo cortado em 5 pedaços iguais. Se o cliente faz um pedido de apenas um pedaço, o valor para esse pedaço é R$ 3,00, enquanto se o cliente faz um pedido de dois pedaços, o valor é R$ 4,00. E assim sucessivamente, conforme demonstra a tabela.

Quantidade de pedaços	1	2	3	4	5
Preço	3,00	4,00	8,00	9,00	12,00

Isabel, estudante dedicada do curso técnico em Informática para Internet, gosta muito dos bolos da padaria e sempre compra um bolo inteiro. Acontece que Aninha percebeu que, dependendo da forma que ela faz o pedido, ela consegue minimizar o valor total do bolo. Por exemplo, se ela comprar o bolo todo fazendo um único pedido de 5 pedaços, o custo total será de R$ 12,00. Entretanto, se ela fizer três pedidos (1 pedaço + 2 pedaços + 2 pedaços), ela consegue levar o bolo todo, mas com o preço total de R$ 11,00 (3,00 + 4,00 + 4,00).

Isabel quer saber qual o valor mínimo que ela pode gastar para comprar um bolo inteiro, dado a quantidade de pedaços que o bolo foi cortado e o valor de cada pedido referente as pedaços.

Entrada

A primeira linha da entrada contém um número inteiro N, que indica a quantidade de pedaços que o bolo foi cortado. A segunda linha contém N valores reais V, que indicam preços de cada possibilidade de pedido.

Restrições

$1 \leq N \leq 10$
$0 < V \leq 1000$

Saída

Seu programa deve imprimir um único número real (com duas casas decimais) do menor valor possível a se pagar pelo bolo todo.

Exemplos de Entradas	Exemplos de Saídas
5 3.0 4.0 8.0 9.0 12.0	11.00
4 3.0 9.0 5.0 13.0	8.00

Problema L
Mensagens codificadas

Por Daniel Corrêa Lobato (IFSP – campus Catanduva)

Arquivo: mensagens.[c|cpp|java\cs\py]

***Timelimit:** 1*

Quando dois computadores se comunicam, eles trocam bits. Esses bits são agrupados para formar símbolos que fazem sentido para os usuários a partir de algum esquema de codificação, como o ASCII. É claro que cada símbolo deve possuir um padrão único de bits, e deve haver padrões suficientes para representar todos os símbolos necessários.

Um dos primeiros códigos utilizados para comunicação de dados foi o código Baudot, que usava 5 bits para representar até 32 símbolos distintos. Entretanto, o esquema de codificação de Baudot utilizava uma artimanha para permitir representar mais símbolos mantendo a quantidade de bits: dois conjuntos de códigos, que podiam ser alternados durante a transmissão a partir de um padrão específico.

A transmissão começava com o primeiro conjunto de símbolos e, no momento no qual o padrão de troca de conjuntos era identificado, a transmissão mudava para o segundo conjunto. O conjunto usado permanecia o mesmo enquanto não fosse identificado, novamente, o padrão de troca de conjuntos.

A tabela abaixo mostra uma versão simplificada da ideia proposta no esquema de codificação do Baudot:

Padrão de bits	Conjunto 1	Conjunto 2
000	A	1
001	B	2
010	Espaço	Espaço
011	C	3
100	D	4
101	Trocar para conjunto 1	Trocar para conjunto 1
110	E	5
111	Trocar para conjunto 2	Trocar para conjunto 2

A mensagem 011000010111011000101100 pode ser decodificada como “CA 31D”.

Você deve implementar um programa que, recebendo os dois conjuntos de símbolos e uma mensagem composta por vários bits, apresente a mensagem correspondente. A transmissão sempre começa com o primeiro conjunto de símbolos.

Entrada

A entrada é composta por duas partes. A primeira parte é composta por duas linhas de 32 caracteres cada uma, contendo os símbolos correspondentes a cada padrão de bits do primeiro e do segundo conjuntos de caracteres respectivamente. O padrão correspondente ao símbolo é dado pela posição do símbolo na linha: o primeiro símbolo vale 00000, o segundo vale 00001, o terceiro vale 00010, e assim sucessivamente. O padrão 11011 sempre é usado para ativar o primeiro conjunto de símbolos, e o padrão 11111 ativa o segundo conjunto de símbolos, e correspondem a espaços em branco nas linhas de entrada. A segunda parte da entrada é composta por uma sequência de “0” e “1” que representam, individualmente, cada bit transmitido. Há, no máximo 100 bits na mensagem a ser decodificada, e a mensagem sempre começa com o primeiro conjunto de caracteres.

Saída

A saída deve ser composta por uma única linha de texto contendo a mensagem decodificada.

Exemplos de Entradas	Exemplos de Saídas
ABCD EFGHIJKLMNOPQRSTUVWXYZ [{(0123 456789!@#$%^&*-_=+,<.>]}) 1001100101011100001100100011010111101110001011100111111101011	SEND MONEY!

Problema M
Misturando tudo

Por Daniel Corrêa Lobato (IFSP – campus Catanduva)
Arquivo: tudo.[c|cpp|java\cs\py]
***Timelimit:** 1*

Na telecomunicação, a multiplexação é uma técnica que consiste na combinação de dois ou mais canais de informação por apenas um meio de transmissão usando um dispositivo chamado multiplexador. O objetivo é compartilhar um recurso escasso (o canal de comunicação) entre diversas fontes de sinal.

Uma das formas de multiplexação é a multiplexação por divisão de tempo. Nela, dois ou mais fluxos de bits são transmitidos aparentemente ao mesmo tempo pelo meio de transmissão, mas eles estão, de fato, se revezando fisicamente no acesso ao canal. O domínio do tempo é dividido em vários slots de tempo de duração constante, um para cada fonte de sinal. Um conjunto de bits da fonte 1 é transmitido durante o slot de tempo 1, da fonte 2 no slot de tempo 2, e assim sucessivamente. Um quadro TDM consiste em um ou mais slots de tempo por fonte, mais alguma informação de sincronização e, eventualmente de detecção de erro. Depois do último slot, o ciclo se reinicia, começando com o segundo conjunto de dados da fonte 1 e assim por diante.

Você deve construir um programa que receba uma determinada quantidade de fluxos de dados e faça a multiplexação desses fluxos, incluindo o código de detecção de erro dentro do quadro, calculado como a soma módulo 256 de todos os dados enviados naquele quadro.

Entrada

A entrada é composta por diversas linhas. A primeira linha contém dois inteiros, N e M ($2 \leq N \leq 10$; $1 \leq M \leq 10$) que indicam, respectivamente, o número de fontes de sinal e o número de bytes por slot. A segunda linha contém um inteiro L ($1 \leq L \leq 1000$; múltiplo de M) que indica o número de bytes total em cada fluxo (todos os fluxos possuem o mesmo tamanho). Em seguida, teremos L * N linhas, cada uma com um inteiro C ($0 \leq C \leq 255$) indicando o conteúdo de cada um dos fluxos, sendo que os L primeiros correspondem aos dados do primeiro fluxo, os L seguintes para o segundo fluxo, e assim sucessivamente.

Saída

A saída deve conter um conjunto de linhas correspondendo ao fluxo multiplexado com os códigos de detecção de erros.

Exemplos de Entradas	Exemplos de Saídas
2 3 6 10 20 30 40 50 60 200 210 220 230 240 250	10 20 30 200 210 220 178 40 50 60 230 240 250 102