

**MARATONA DE PROGRAMAÇÃO 2020**



# **CADERNO DE PROBLEMAS**

São Paulo – Brasil

**Novembro, 2020**

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Instruções

- 1) Este caderno contém 10 problemas (A-J). As páginas estão numeradas de 1 a 17, não contando a página de rosto. Verifique se o caderno está completo antes de começar.
- 2) Em todos os problemas, os programas deverão ler a entrada padrão para recebimento de dados externos e mostrar na saída padrão os resultados esperados.
- 3) Em todos os problemas, em que o final da entrada não esteja especificado no texto do problema, deverá ser considerado o final das entradas.
- 4) **Excepcionalmente nesta edição, será permitido** a utilização de código pronto. Os códigos poderão ser consultados de anotações e publicações em geral, Internet e armazenamento local.
- 5) **Excepcionalmente nesta edição, será permitido** a utilização da Internet além do acesso e utilização da plataforma oficial da competição, isto é, o software BOCA.
- 6) Qualquer desrespeito às normas de restrições acima, o competidor poderá ser desclassificado;
- 7) Qualquer dúvida envie uma *Clarification*.

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Dicas para Iniciantes

Aqui vão algumas dicas rápidas para quem não está acostumado com os tipos de problemas da maratona e das olimpíadas de informática em geral:

- Todo problema tem uma entrada exemplo e a saída esperada para aquela entrada. Você deve sempre lembrar que essa entrada **não é nem de perto** a entrada que os juízes irão usar para testar seu programa. Lembre-se sempre de que é apenas um exemplo. Após fazer o programa funcionar para os exemplos, teste sempre condições extremas e de contorno (verifique os limites do problema, qual os valores mínimo e máximo que as variáveis podem atingir etc.).
- Evite mandar programas precipitadamente. É bem melhor gastar 5 minutos testando o programa para ter certeza de que funciona para as mais diversas entradas do que receber uma mensagem de erro dos juízes (que acarreta uma penalização de 10 minutos).
- A maioria dos problemas sempre tem um "truque" escondido. Por exemplo, suponha um problema trivial de calcular fatorial. Você se lembrou de tratar o problema quando a entrada é zero? Sempre procure pensar no que o juiz deveria estar pensando para fazer a entrada. Afinal, o seu programa deve funcionar corretamente para qualquer entrada.
- Se o seu algoritmo lidar com busca, procure cortar os nós ao máximo. Quanto menos nós expandidos, mais eficiente será seu programa. Problemas de busca são, em geral, críticos com relação ao tempo, e um descuido pode acarretar estouro do tempo limite!
- A maratona é um torneio de criação e implementação de algoritmos. Isto significa que você nunca vai precisar se preocupar com coisas do tipo: interface com o usuário, reusabilidade do código etc. Aliás, é bem melhor usar nomes sugestivos de variáveis do que comentários.
- Evite usar as ferramentas de debug, elas consomem muito tempo. Aliás, deve-se evitar "debugar" o problema no computador, afinal, o tempo é curto. Enquanto você estiver "debugando" o programa na tela do computador, pode estar deixando de fazer outro mais fácil. Se for indispensável o "debug" em "tempo real", procure fazê-lo usando "print" de variáveis e técnicas do tipo. Às vezes, breakpoints também podem ser úteis.
- **Ninguém** vai analisar seu código, portanto não interessa se ele está elegante ou não. Se você tiver uma solução "feia", porém eficiente, essa é a que se deve usar.
- Nem sempre um algoritmo polinomial pode ser viável. Suponha que você tenha implementado um algoritmo  $O(n^3)$  para um determinado problema. Mas e se não puder assumir valores até, digamos, 1000? Com quase toda certeza seu problema irá estourar o limite de tempo. Por isso, sempre preste atenção não só à complexidade do seu algoritmo, como à viabilidade de sua implementação.
- Não há nada pior do que gastar muito tempo fazendo e implementando um algoritmo para, depois, descobrir que ele está errado. Numa maratona, esse erro é fatal. Por isso, apesar de a pressa ser necessária, procure sempre certificar-se da corretude do algoritmo **ANTES** de implementá-lo!
- Verifique sua saída, para não deixar espaços em branco ao final de cada linha. Isso será considerado errado na validação.

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Problema A

#### Anagrama

*Arquivo fonte: Anagrama.{ c | cc | java | py3 }*

*Autor: Prof. Henrique Louro (Etec de Caraguatatuba)*

#### Tarefa

Anagrama é um jogo de palavras que utiliza a transposição ou rearranjo de letras de uma palavra ou frase, com o intuito de formar outras palavras com ou sem sentido. As letras na palavra formada devem constar na mesma quantidade que na palavra original. Neste problema permitiremos utilizar menos letras para formar outras palavras, com no mínimo três letras da palavra original. Assim sendo, a palavra “pata” não pode ser considerada um anagrama da palavra “paletó” pois essa palavra não possui duas vogais “a”. Já a palavra “pato” pode ser considerada um anagrama de “paletó”.

Sua tarefa é que dada uma certa palavra, e um conjunto de palavras provavelmente formadas a partir das suas letras, verificar quais podem ser consideradas um anagrama da palavra original.

#### Entrada

O arquivo de entrada terá vários casos de teste. Cada caso estará em uma linha da entrada e será composto por uma palavra original e um conjunto de  $N$  ( $1 \leq N \leq 30$ ) palavras à sua frente, separadas por um espaço, com no mínimo 3 letras e no máximo a quantidade de letras da palavra original. Os acentos e a cedilha foram retirados para facilitar a resolução do problema. As entradas deverão ser lidas da entrada padrão. Uma linha com apenas um número 0 encerra as entradas.

#### Saída

Para cada caso de teste, seu programa deverá informar a quantidade  $Q$  ( $0 \leq Q \leq 30$ ) de anagramas encontrados no conjunto de palavras à frente da palavra original. As saídas deverão ser mostradas na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
paletó pela pate pata pato pelo pote paleta	5
policia cao ali pole capo cola paca pelica	4
casa saca caso saco asa	2
sacola saco cola calo asco sala asa cao	7
escola cola calo colo seco seca ola eco	6
0	

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Problema B

#### Mega-Sena

*Arquivo fonte: Megasena.{ c | cc | java | py3 }*

*Autor: Prof. Henrique Louro (Etec de Caraguatatuba)*

#### Tarefa

Jogos de loteria são eventos aleatórios que envolvem combinações. A Mega-Sena, por exemplo, consiste na combinação de seis números selecionados de uma sequência de 1 a 60. O jogador pode combinar de 6 a 15 números numa única aposta. A Caixa Econômica Federal (CEF) sorteia, a cada concurso, apenas seis números, e cada número sorteado é retirado do jogo. Ou seja, na combinação da Mega-Sena não é possível haver números repetidos. Os prêmios são distribuídos entre quem faz a quadra (acertar 4 números), a quina (acertar 5 números) ou a sena (acertar 6 números), qualquer que seja a ordem do sorteio. Segundo a CEF, a chance de acertar a sena jogando apenas 6 números é de 1 para 50.063.860.

João Bolão é um apostador inveterado. Não perde um concurso. Tentando melhorar suas chances de acerto, conseguiu as informações de todos os concursos já realizados pela CEF, até 03/10/2020. Ele gostaria de saber, quais foram os números mais sorteados em todos esses concursos. Sabendo que você é um excelente programador, pediu sua ajuda para computar quais seriam esses números e quantas vezes isso aconteceu com cada um deles.

#### Entrada

Consiste em vários casos de teste, sendo uma linha para cada caso. Cada linha contém a informação de um único concurso, começando com o número (X) do concurso, onde  $1 \leq X \leq 2305$ , a data em que foi realizado, e a sequência das 6 dezenas (D) sorteadas, onde  $1 \leq D \leq 60$ . Todos os dados separados por um espaço. A entrada deverá ser lida da entrada padrão. A entrada termina com uma linha com o número 0, apenas para indicar o final do arquivo.

#### Saída

Deverá ser uma lista por ordem da quantidade de vezes que cada dezena foi sorteada, da maior para a menor, de acordo com a lista de concursos fornecida na entrada. Se houver empate na quantidade de vezes que uma ou mais dezenas foram sorteadas, deverão ser ordenadas da menor para a maior dezena. Cada linha na saída deverá conter a posição ordinal de sua colocação (P), onde  $1 \leq P \leq 60$ , a dezena (D) e a quantidade (Q) de vezes que foi sorteada, onde  $0 \leq Q \leq 2305$ . Todas as informações deverão ser separadas por um espaço. A saída deverá ser mostrada na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

1;11/03/1996;41;5;4;52;30;33	1 1 2
2;18/03/1996;9;39;37;49;43;41	2 6 2
3;25/03/1996;36;30;10;11;29;47	3 30 2
4;01/04/1996;6;59;42;27;1;5	4 41 2
5;08/04/1996;1;19;46;6;16;2	5 2 1
0	6 4 1
	7 5 1
	8 5 1
	9 9 1
	10 10 1
	11 11 1
	12 16 1
	13 19 1
	14 27 1
	15 29 1
	16 33 1
	17 36 1
	18 37 1
	19 39 1
	20 42 1
	21 43 1
	22 46 1
	23 47 1
	24 49 1
	25 52 1
	26 59 1

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Problema C

#### Calculadora Romana

*Arquivo fonte: Romana.{ c | cc | java | py3 }*

#### Tarefa

Os números romanos foram utilizados na antiguidade, durante o império romano. Eram representados conforme a tabela abaixo. O objetivo deste exercício é programar uma calculadora que some dois ou mais números romanos emitindo o resultado da operação, também em números romanos.

Símbolo	Nome	Valor
I	unus	1 (um)
V	quinque	5 (cinco)
X	decem	10 (dez)
L	quingenta	50 (cinquenta)
C	centum	100 (cem)
D	quingenti	500 (quinhentos)
M	mille	1000 (mil)

Vale lembrar que uma unidade inferior antes da seguinte, diminui seu valor da próxima. Por exemplo: IV = 4 => I = 1 e V = 5, assim teremos V menos I, o que resultará em 4.

Já uma unidade inferior à frente da seguinte, aumenta seu valor na próxima. Por exemplo: VI = 6 => V = 5 e I = 1, assim teremos V mais I, o que resultará em 6.

#### Entrada

A entrada é composta por diversos casos de testes, cada um em uma linha. Cada uma das linhas pode conter dois ou mais números romanos, separados pelo sinal de "+". A maior soma possível dos valores dados em uma linha será 3999. A entrada deverá ser lida da entrada padrão. O final da entrada é marcado pelo número decimal 0.

#### Saída

Para cada caso de teste imprima uma linha contendo um único número romano indicando a soma dos números dados na linha correspondente na entrada. A saída deverá ser mostrada na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
X+XV+IV DXX+MVI+V VII+CXI D+D+V+V XX+XV+XIV 0	XXIX MDXXXI CXVIII MX XLIX

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Problema D

#### Receita de Bolo

*Arquivo fonte: Receita.{ c | cc | java | py3 }*

Autores: Prof. Hamilton Machiti (ETEC São Bernardo do Campo) e

Prof. Henrique Louro (ETEC de Caraguatatuba)

#### Tarefa

Receita é um termo que tem vários conceitos e significados diversos dentro do que se pode definir ou conceituar. É uma palavra amplamente difundida na língua portuguesa e é de fácil explicação no âmbito global dos seus significados. Um dos significados ou conceitos do termo receita é a fórmula escrita e detalhada para se fazer um alimento, ou seja, uma fórmula de fabricação com todos os detalhes descritos para que se possa obter um prato ou uma bebida específica, seguindo uma série de instruções que resultam finalmente no que se vai degustar.

Joaquim Mario da Paz é dono de uma padaria especializada em fabricar bolos. Mas ele está com um grande problema: não consegue controlar quanto gasta de ingredientes para fazer todos os bolos que vende em um determinado período. Sabendo que você está estudando programação de computadores, pediu que fizesse um programa que o ajudasse nessa tarefa. Dadas as três receitas dos bolos que fabrica e a quantidade fabricada diariamente de cada um, seu programa deverá informar a quantidade geral de ingredientes utilizados para fabricá-los.

RECEITAS DE BOLOS				
Item	Ingredientes	Milho	Coco	Chocolate
		Quantidade	Quantidade	Quantidade
1	Milho verde	200g	-	-
2	Óleo vegetal	200ml	120ml	240ml
3	Açúcar	250g	360g	160g
4	Fubá	200g	-	-
5	Ovos	4	4	2
6	Farinha de trigo	15g	240g	240g
7	Coco ralado	15g	100g	-
8	Fermento em pó	5g	10g	15g
9	Leite de coco	-	200ml	-
10	Chocolate em pó	-	-	90g
11	Leite	-	-	240ml



## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Entrada

A entrada é composta de um único caso de teste. A primeira linha contém um inteiro  $D$  que representa a quantidade de dias de fabricação ( $1 \leq D \leq 180$ ). As  $D$  linhas seguintes trazem três números inteiros que representam as quantidades de bolos de milho (BM), de coco (BC) e de chocolate (BCH) fabricados por dia ( $1 \leq BM, BC, BCH \leq 1000$ ). Todos os dados separados por um espaço. As entradas deverão ser lidas da entrada padrão e encerram-se com uma linha contendo o número 0.

### Saída

Como saída, seu programa terá que mostrar 11 linhas, contendo o número do item, a descrição (com acentos e cedilha suprimidos para compatibilidade dos sistemas) e a quantidade necessária de cada um dos ingredientes para realizar a fabricação dos bolos, na ordem em que aparecem na receita e separados por um espaço. As quantidades deverão ser seguidas das suas unidades de medidas, sem espaço, convertidas em Kg, para peso e L, para volume, quando houver. Havendo casas decimais nas quantidades, estas deverão ser desprezadas. As saídas deverão ser mostradas na saída padrão.

Exemplo de Entrada	Saída para o exemplo de entrada
5 100 200 300 150 300 200 500 250 400 200 400 300 250 250 250 0	1 Milho verde 240Kg 2 Oleo vegetal 756L 3 Acucar 1036Kg 4 Fuba 240Kg 5 Ovos 13300 6 Farinha de trigo 702Kg 7 Coco ralado 158Kg 8 Fermento em po 41Kg 9 Leite de coco 280L 10 Chocolate em po 130Kg 11 Leite 348L

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Problema E

#### Moeda Blemflarck

*Arquivo fonte: Moeda.{ c | cc | java | py3 }*

*Autor: Prof. Hamilton Machiti (Etec de São Bernardo do Campo)*

#### Tarefa

O ano é 2122, os planetas que fazem parte da federação galáctica enviaram seus representantes para uma reunião no planeta Gazorpazorp. Nesse encontro foram discutidos assuntos econômicos, entre eles a criação de uma moeda unificada para todos os membros da federação, surgindo então a moeda Blemflarck (B\$). Ficou acertado que para a Blemflarck, serão emitidas cédulas dos seguintes valores: B\$ 50,00, B\$ 10,00, B\$ 5,00 e B\$ 1,00. Enquanto que para atender aos cidadãos galácticos, serão implantados caixas eletrônicos em todos os planetas da federação e você foi contratado para desenvolver a programação destes equipamentos.

Os caixas eletrônicos operam com todos os valores de notas da Blemflarck (B\$ 50,00, B\$ 10,00, B\$ 5,00 e B\$ 1,00), mantendo um estoque de cada uma em seu cofre interno. Os cidadãos da federação galáctica dirigem-se até o caixa eletrônico mais próximo e após as devidas credenciais apresentadas, efetuam a retirada de um número inteiro de Blemflarcks.

Sua tarefa é produzir um programa que, dado o valor de Blemflarcks informado pelo cidadão, determine a quantidade de cada uma das notas para totalizar esse valor, de modo a minimizar o número de cédulas entregues.

Por exemplo, se o cidadão deseja retirar B\$ 10,00, basta entregar uma única nota de dez Blemflarcks. Em um segundo exemplo, se o galáctico deseja retirar B\$ 63,00, é necessário entregar uma nota de B\$ 50,00, uma de B\$ 10,00 e três de B\$ 1,00.

#### Entrada

A entrada terá vários casos de teste. Começa com um inteiro  $T$  ( $1 \leq T \leq 500$ ) que indica a quantidade de casos de teste que seguem. Cada caso de teste consiste em uma linha com um único valor inteiro  $N$  ( $1 \leq N \leq 10000$ ), que indica o valor solicitado pelo cidadão. As entradas deverão ser lidas da entrada padrão.

#### Saída

Para cada caso de teste, a saída de seu programa deve produzir uma única linha, apresentando o texto Caso de Teste #: (com  $1 \leq \# \leq 500$ ), seguido de um espaço em branco. A seguir, na mesma linha, deve apresentar uma sequência de quatro números inteiros: O, P, Q e R, que representam o resultado encontrado pelo seu programa: O indica o total de cédulas de B\$ 50,00, P indica o número de cédulas de B\$ 10,00, Q indica o número de cédulas de B\$ 5,00 e R indica o número de cédulas de B\$ 1,00. As saídas deverão ser mostradas na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
4 462 9638 347 28	Caso de Teste 1: 9 1 0 2 Caso de Teste 2: 192 3 1 3 Caso de Teste 3: 6 4 1 2 Caso de Teste 4: 0 2 1 3

## CADERNO DE PROBLEMAS

### MARATONA DE PROGRAMAÇÃO 2020

#### Problema F

##### CPF

*Arquivo fonte: Cpf.{ c | cc | java | py3 }*

*Autor: Prof. Henrique Louro*

##### Tarefa

Segundo o site da Caixa Econômica Federal, o **Cadastro de Pessoa Física (CPF)** é o documento que identifica o contribuinte perante a Receita Federal. Cada contribuinte pessoa física possui um Cartão **CPF**, ou simplesmente **CPF**, que comprova esse cadastro. Ele contém um número identificador que não muda. É composto por 11 dígitos de 0 a 9, sendo que os dois últimos dígitos são chamados de dígitos verificadores, que validam os 9 primeiros.

Existe uma fórmula para verificar se o CPF é válido ou não. A fórmula verifica se os dígitos verificadores são resultado dos 9 primeiros dígitos do CPF. É bom lembrar que essa fórmula só verifica se os números do CPF são válidos e não se o CPF pertence à alguma pessoa.

Sendo assim, sua tarefa é pesquisar na Internet tal fórmula de validação, implementá-la como função no seu código e validar uma lista de CPFs recebida. Lembre-se que essa função deve fazer parte do seu código e não pode ser um objeto externo, pois afinal você só pode enviar um único código por problema.

##### Entrada

A entrada terá vários casos de teste. Cada caso, será composto por um conjunto de 11 números de 0 a 9 cada, que representam os números de um CPF. Cada caso de teste estará em uma linha da entrada. As entradas deverão ser lidas da entrada padrão. Uma linha com apenas um número 0 encerra as entradas.

##### Saída

Para cada caso de teste, seu programa deverá validar os 11 dígitos, informando se se trata de um CPF válido ou não. Para um CPF válido deverá ser mostrada a palavra "Sim". Já para os inválidos deverá ser mostrada a palavra "Não". As saídas deverão ser mostradas na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
00291706053	Sim
12345678910	Não
27312100872	Sim
12121212121	Não
00221201068	Sim
33333333333	Não
83204067815	Sim
96969696969	Não
00347302068	Sim
10987654321	Não
60680199420	Sim
00150002068	Sim
0	

(\*) **Observação:** Os CPFs utilizados nesse problema, são todos gerados aleatoriamente. Qualquer CPF válido existente, não passa de mera coincidência.

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Problema G

#### Média Final

*Arquivo fonte: Media.{ c | cc | java | py3 }*

*Autores: Prof. Hamilton Machiti*

#### Tarefa

Há alguns anos, a Universidade Costa Lima passou a utilizar um sistema de avaliação de alunos único para todos os cursos. A ideia é que a nota final de um aluno seja composta por um número pré-determinado de notas que devem ser obtidas por meio de instrumentos de avaliação diferentes. Os instrumentos são os seguintes: D1, D2, D3, A1 e A2. Em geral, os três primeiros são atividades realizadas cada uma em um único dia, como uma prova escrita. Os dois últimos costumam ser compostos por notas obtidas em pequenas atividades realizadas ao longo do semestre, ou seja, permitem que os alunos sejam avaliados de forma continuada. Cada um dos cinco instrumentos vale 20 pontos. Para ser aprovado, a média final do aluno deve ser de, pelo menos, 60. Um professor da Costa Lima, neste semestre, irá calcular a média final (MF) de seus alunos da seguinte forma:

$$MF = D1 + D2 + D3 + (A11 + A12) / 2 + (A21 + A22 + A23) / 3$$

onde  $D1 \leq 20$ ,  $D2 \leq 20$ ,  $D3 \leq 20$ ,  $A1i (1 \leq i \leq 2) \leq 20$ ,  $A2i (1 \leq i \leq 3) \leq 20$

Note que, para cada aluno, haverá 8 notas envolvidas no cálculo da sua média final. O professor deseja que esse valor seja truncado. Ou seja, deseja desprezar todas as casas decimais.

Dadas oito notas de um aluno, você pode ajudar o professor a encontrar a média final?

#### Entrada

A entrada terá vários casos de teste. Cada caso, estará em uma linha e será composto por oito números inteiros separados por um espaço em branco. Eles aparecerão na seguinte ordem: D1, D2, D3, A11, A12, A21, A22, A23, separados por um espaço. As entradas deverão ser lidas da entrada padrão. Uma linha com apenas um número 0 encerra as entradas.

#### Saída

Para cada caso de teste, seu programa deverá exibir "Aluno aprovado com nota: N" ( $0 \leq N \leq 100$ ) caso a média final calculada seja maior ou igual a 60 e "Aluno reprovado com nota N", caso contrário. No lugar de N, o programa deve exibir a média final obtida. As aspas não devem ser exibidas. As saídas deverão ser mostradas na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
17 20 14 10 20 0 0 20 0 2 20 12 15 10 10 20 20 10 16 10 10 8 6 5 16 12 14 8 11 9 7 5 0	Aluno aprovado com nota: 72 Aluno reprovado com nota: 48 Aluno aprovado com nota: 62 Aluno reprovado com nota: 58

## CADERNO DE PROBLEMAS

### MARATONA DE PROGRAMAÇÃO 2020

#### Problema H

##### Batalha Naval

*Arquivo fonte: Naval.{ c | cc | java | py3 }*

*Autor: Prof. Henrique Louro*

#### Tarefa

Batalha naval é um jogo de tabuleiro de dois jogadores, no qual eles têm que adivinhar em que quadrados estão os navios do oponente. O jogo foi originalmente jogado com lápis e papel. Seu objetivo é localizar e “afundar” os barcos do oponente. Ganha quem afundar todos os navios adversários primeiro.

O jogo original é jogado em dois quadros quadriculados para cada jogador — um que representa a disposição dos barcos do jogador, e outro que representa a do oponente. Cada quadro quadriculado tem suas linhas na horizontal identificadas por números e colunas na vertical identificadas por letras. Em cada quadro quadriculado o jogador coloca os seus navios e registra os tiros do oponente.

Antes do início do jogo, cada jogador coloca os seus navios nos quadros, alinhados horizontalmente ou verticalmente. O número de navios permitidos é igual para ambos jogadores e os navios não podem se sobrepor.

Após os navios terem sido posicionados, o jogo continua numa série de turnos. Em cada turno, um jogador dá um “tiro” dizendo um quadrado, o qual é identificado pela letra e número, no quadro do oponente. Se houver um navio nesse quadrado, é colocada uma marca vermelha, e informado que o tiro acertou um alvo. Se, no entanto, não houver navio é colocada uma marca branca e informado que o tiro acertou a água.

Os tipos de navios são: porta-aviões (cinco quadrados), navios-tanque (quatro quadrados), contratorpedeiros (três quadrados) e submarinos (dois quadrados). Vale notar que os quadrados que compõem um navio devem estar conectados e em linha reta. Numa das variações deste jogo, os quadros são de dimensão 10x10, e o número de navios são: 1, 2, 3 e 4 respectivamente.

Para esse problema fizemos algumas modificações. As letras foram substituídas por números. Você receberá uma matriz  $M \times M$  com os navios já posicionados, e uma sequência de coordenadas  $L \times C$ , indicando os tiros dados. Você deverá informar, para cada tiro se acertou um alvo ou a água.

#### Entrada

A entrada terá um único caso de teste. Será composto por uma linha com 2 inteiros separados por espaços,  $M$  e  $T$ , onde  $M$  ( $10 \leq M \leq 100$ ) que é a quantidade de linhas e colunas da matriz onde os navios estarão posicionados previamente, e  $T$  ( $1 \leq T \leq 500$ ) que é a quantidade de “tiros” dados. Em seguida  $M$  linhas compostas de  $M$  caracteres “A” e “X”, sem separação, onde A indica que a posição é vazia ou água e X uma parte de um navio, conforme formação e quantidades informados no texto da tarefa. Na sequência  $T$  linhas indicando as coordenadas  $L \times C$  de cada “tiro” dado, sendo  $L$  ( $1 \leq L \leq M$ ) a linha e  $C$  ( $1 \leq C \leq M$ ) a coluna na matriz  $M \times M$  dada. As entradas deverão ser lidas da entrada padrão.

#### Saída

Para cada “tiro” na entrada, seu programa deverá informar em uma linha na saída a palavra “acertou” caso encontre o caractere “X” na posição informada, indicando que o tiro acertou um navio, ou “agua” (sem acento) caso o caractere encontrado seja um “A” indicando que o tiro caiu na água. As saídas deverão ser mostradas na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
10 10 XAAAAAAAAA	agua acertou

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

XAXXXXXAAA XAAAAAAAAA XAXXAXXXA XAXXAAAAA AAAXAAAXX AAAXAAAAA XAAAXXAAA XAAAAAXXX XAXXAXAAA	agua acertou acertou agua agua agua acertou agua
1 2 2 1 3 4 9 7 10 4 3 5 6 6 8 2 5 4 4 10	

## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Problema I

#### Palíndromo

*Arquivo fonte: Palin.{ c | cc | java | py3 }*

*Autor: Prof. Henrique Louro*

#### Tarefa

Dizemos que um número é palíndromo quando seus algarismos lidos da esquerda para direita e da direita para esquerda apresentam o mesmo número. Por exemplo, o número 75457 é um palíndromo.

É claro que essa propriedade depende da base em que o número é expresso. O número 17 não é um palíndromo na base 10, porém representado na base 2 (10001) será um palíndromo.

O objetivo deste problema é verificar se dado um conjunto de números inteiros aleatórios na base 10, representam números palíndromos na base 2.

#### Entrada

A entrada terá vários casos de teste. Cada caso estará em uma linha, que será composta de um número inteiro aleatório  $N$  ( $1 \leq N \leq 50000$ ), na base decimal. As entradas deverão ser lidas da entrada padrão. As entradas terminam com um número 0, que não deverá ser verificado.

#### Saída

Para cada caso de teste, seu programa deverá mostrar na saída padrão, a letra V (Verdadeiro) quando o número convertido representa um palíndromo na base 2. Caso não represente, deverá ser mostrada a letra F (Falso). As saídas deverão ser mostradas na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
82	F
17	V
85	V
754	F
231	V
907	F
0	



## CADERNO DE PROBLEMAS MARATONA DE PROGRAMAÇÃO 2020

### Problema J

#### Peça Perdida

*Arquivo fonte: Peca.{ c | cc | java | py3 }*

*Autor: Prof. Henrique Louro*

#### Tarefa

Joãozinho adora quebra-cabeças. Essa é sua brincadeira favorita. O grande problema, é que às vezes o jogo vem com uma peça faltando. Isso irrita bastante o pobre menino, que tem que descobrir qual peça está faltando e solicitar uma de reposição ao fabricante do jogo. Sabendo que o quebra-cabeças tem  $X$  peças, numeradas sequencialmente de 1 a  $X$  ( $X \in \mathbb{N}$ ), e que exatamente uma está faltando, ajude Joãozinho a saber qual peça ele tem de pedir.

Escreva um programa que, dado um inteiro  $X$  e  $X - 1$  inteiros numerados de 1 a  $X$ , descubra qual número está faltando.

#### Entrada

A entrada contém vários casos de testes. Cada caso é composto por 2 linhas. A primeira contém um inteiro  $X$  ( $3 \leq X \leq 100$ ). A segunda contém  $X - 1$  inteiros, não necessariamente em ordem sequencial, numerados de 1 a  $X$  (sem repetições), separados por um espaço. As entradas deverão ser lidas da entrada padrão. Uma linha com apenas um número 0 encerra as entradas.

#### Saída

Para cada caso de teste, seu programa deverá mostrar uma linha contendo o número inteiro que está faltando no conjunto de números inteiros dado. As saídas deverão ser mostradas na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
3	2
3 1	4
5	1
1 2 3 5	
4	
2 4 3	
0	



**CADERNO DE PROBLEMAS  
MARATONA DE PROGRAMAÇÃO 2020**

**BOA SORTE!**



**Apoio:**

