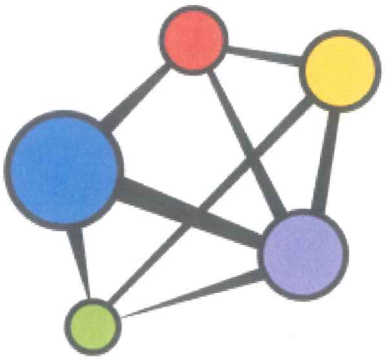


9ª Maratona de Programação

Facens 2013

CADERNO DE PROBLEMAS

PROBLEMA A: PRIMOS SÓ DÃO PROBLEMAS

Arquivo: `primos.[cpp/c/java]`

Cor: laranja

Limite de tempo: 2s

Descrição do problema

Joãozinho precisa contar os primos e você, um de seus primos favoritos, decidiu ajuda-lo. Ele precisa que você crie um programa que conte os números primos dentro de um determinado intervalo.

Entrada

A entrada é constituída de vários casos de teste. Cada caso de teste possui dois valores A e B que correspondem ao valor inicial e valor final do intervalo, respectivamente. O final de entrada é indicado por A = B = 0. Considere 2 <= A < B <= 1000000.

Saída

A saída deverá ser composta por uma linha para cada entrada contendo a quantidade de números primos no intervalo de A a B.

Exemplos de testes

Entrada	Saída
2 10 15 27 70 72 0 0	4 3 1

PROBLEMA B: NÚMEROS ABUNDANTES

Arquivo: `abunda.[cpp/c/java]`

Cor: branco

Limite de tempo: 1s

Descrição do problema

Podemos classificar os números de acordo com suas propriedades como amigos, primos, perfeitos, defectivos, abundantes, etc.
Sua equipe certamente já precisou implementar um código que busca pelas propriedades de um número para poder classifica-lo.
A tarefa destinada a sua equipe é encontrar todos os números abundantes de um dado intervalo. Um número natural que não é perfeito nem defectivo é dito um número abundante, ou seja, a soma de seus divisores próprios é superior a ele.

Entrada

A entrada é constituída de vários casos de teste. Cada caso possui dois números inteiros que representam o intervalo. O programa deverá ser encerrado quando informado dois números iguais, ou seja, não há intervalo. Os números estão no intervalo de zero a 1000.

Saída

A saída serácomposta por uma sequência de números inteiros abundantes ou por um traço (-) caso não exista números abundantes.

Exemplos de testes

Entrada	Saída
2 10	-
27 15	18 20 24
70 72	70 72
1 1	

PROBLEMA C: CANCELA

Arquivo: `cancela.[cpp/c/java]`

Cor: verde escuro

Limite de tempo: 1s

Descrição do problema

A FACENS instalou, recentemente, um sistema de cancela automática e distribuiu para cada aluno e funcionário um TAG RFID com um identificador único. Agora o professor Martelo precisa de um relatório que apresente todos os carros que ainda estão presentes no estacionamento ordenados pelo horário de entrada dos mesmos. Ele pediu que você criasse um programa para gerar esse relatório a partir da lista de carros que entraram e sairam do estacionamento. Lembramos que um carro não entra novamente sem antes ter saído e nenhum carro sai sem antes ter entrado.

Entrada

A entrada é constituída de vários casos de teste. Cada caso de teste possui um número N que corresponde ao número de carros que entraram e saíram do estacionamento seguido por N linhas, cada uma contendo dois valores I e T que correspondem ao identificador do TAG e o tipo da cancela pelo qual o carro passou (E para entrada e S para saída), respectivamente. Considere que essas linhas estão ordenadas pelo horário de entrada dos carros. O final de entrada é indicado por N = 0.

$1 <= N <= 100$
$100 <= I <= 200$
$T \in \{E, S\}$

Saída

A saída deverá ser composta pela lista de identificadores dos carros que continuam no estacionamento.

Exemplos de testes

Entrada	Saída
5 101 E 102 E 101 S 102 S 101 E 7 109 E 105 E 103 E 101 E 111 E 110 E 105 S 0	101 109 103 101 111 110

PROBLEMA D: ÁRVORE DE NATAL

Arquivo: `natal.[cpp/c/java]`

Cor: vermelho

Limite de tempo: 1s

Descrição do problema

As tradições de Natal sempre encantaram Maria, que neste ano de 2013 pretende decorar sua árvore de Natal. A árvore de natal é mais antiga que o próprio sentido do Natal. Aproximadamente dois milênios antes do nascimento de Cristo, os povos indo-europeus já utilizavam árvores com um fim religioso, pois acreditavam que elas eram uma expressão da energia de fertilidade da Mãe Natureza.
A árvore de Natal é uma das mais populares tradições associadas com a celebração do Natal. É normalmente uma árvore conífera de folhas perenes. Como parte da tradição, enfeita-se a árvore com bolas coloridas e outros adornos natalinos.
Maria pretende decorar sua árvore, que terá características bem particulares, com bolas numeradas, ou seja, cada bola contém um número inteiro e em cada galho da árvore só podem ser inseridas duas bolas, uma na esquerda e outra na direita.
Para organizar sua árvore Maria decidiu que a primeira bola que escolher (aleatoriamente) será colocada na ponta da árvoree a partir dela serão colocadas bolas na esquerda e na direita de tal forma que a direita da árvore contenham apenas bolas com números maiores ou iguais ao valor da bola acima (pai), e da mesma forma, o lado esquerdo da árvore contém apenas bolas com valores menores.
Como Maria tem ficado muito confusa na hora de montar sua árvore ela contratou sua equipe para escrever um programa em que dado uma sequência de números inteiros, construa a árvore seguindo suas regras.

Entrada

A entrada é constituída de vários casos de teste. Cada caso possui uma sequência de dez números inteiros que representam as bolas numeradas. O programa deverá ser encerrado quando o primeiro valor da sequência de dez números tiver o valor zero.

Saída

A saída serácomposta pela sequência de números inteiros que representam os valores das bolas a cada nível da árvore organizada de Maria.

Exemplos de testes

Entrada	Saída
5 6 2 4 9 1 4 5 2 9 -1 2 3 -4 -5 -6 7 9 2 123 78 4 23 56 7 98 4 2 3 -6 0 1 2 3 4 5 6 7 8 9	5 2 6 1 4 5 9 2 4 9 -1 -4 2 -5 3 -6 2 7 9 123 78 4 98 2 23 -6 3 7 56 4

PROBLEMA E: TRUCO

Arquivo: `truco.[cpp/c/java]`

Cor: preto

Limite de tempo: 1s

Descrição do problema

Você faz parte de uma equipe de desenvolvimento de jogos, a Bene Soft. Sua tarefa é criar um algoritmo que determine qual o jogador vencedor em uma rodada de Truco, na qual apenas 2 pessoas estão jogando. Para isso, seu querido chefe Tonhão explicou as regras do jogo:

- Cartas utilizadas: A (ás), 2, 3, 4, 5, 6, 7, Q (dama), J (valete), K (rei)
- Naipes: O (ouros), E (espadas), C (copas), P (paus)

Cada carta possui um valor. A carta de menor valor é o 4, e a de maior valor é o 3. Portanto, a ordem das cartas baseada em seu valor é:

4, 5, 6, 7, Q, J, K, A, 2, 3

No Truco existe o chamado "Tombo", que consiste em uma carta de referência para indicar quais são as "manilhas" da rodada. As manilhas serão as cartas cujo valor é 1 acima do valor da carta do tombo, por exemplo: se o tombo for 3P (três de paus), as manilhas serão 4O (quatro de ouros, manilha de ouros), 4E (quatro de espadas, manilha de espadas), 4C (quatro de copas, manilha de copas) e 4P (quatro de paus, manilha de paus ou "zap"). As cartas que são manilhas perdem seu valor original e passam a ser as cartas de maior valor do jogo. Sendo assim, a ordem final das cartas do Truco é:

4, 5, 6, 7, Q, J, K, A, 2, 3, Manilha ouros, Manilha espadas, Manilha copas, Manilha paus

Basicamente, um jogador deve jogar cartas melhores que a do adversário para vencer uma rodada. Uma rodada é constituída de, no máximo, 3 "mãos". Em cada mão, os jogadores jogam 1 carta cada um e quem jogar a melhor carta vence a mão. Um jogador deve vencer 2 mãos para ser o vencedor da rodada (quando não há empate).
Em caso de empate na 1ª mão, o jogador que vencer a próxima mão vence a rodada. Se houver empate na 1ª e na 2ª mão, a terceira mão decide o vencedor; caso haja empate nas 3 mãos, o jogador que começou a rodada vence. Considere que é sempre o Jogador 1 quem começa a rodada.
Em caso de empate em alguma mão diferente da 1ª, o jogador que venceu a 1ª mão vence a rodada.

Entrada

A entrada é constituída de vários casos de teste. A primeira linha de um caso de teste contém 3 caracteres T, N e M que correspondem, respectivamente, ao número da carta do Tombo, naipe da carta do Tombo e número de mãos que a rodada teve. As próximas M linhas de um caso de teste possuem 4 caracteres A, B, C e D, onde A e B simbolizam respectivamente número e naipe da carta do Jogador 1, e C e D simbolizam respectivamente número e naipe da carta jogada pelo Jogador 2. O final da entrada é indicado por T = N = M = 0 (zero).

Restrições

T, A e C pertencem a { 4, 5, 6, 7, Q, J, K, A, 2, 3 }
N, B e D pertencem a { O, E, C, P }
2 <= M <= 3

Saída

A saída deverá ser composta, para cada caso de teste, por uma linha contendo o jogador que venceu a rodada.

Exemplos de testes

Entrada	Saída
7 E 3 Q O 4 P K E A C 3 O 3 P 4 O 2 3 C 3 P 5 O 5 E 2 C 3 4 O 4 P 5 E 5 C 6 P 6 O 0 0 0	Jogador 1 ganha Jogador 2 ganha Jogador 1 ganha

PROBLEMA F: CÓDIGO 39

Arquivo: `codigo39.[cpp/c/java]`

Cor: azul claro

Limite de tempo: 1s

Descrição do problema

Códigos de barras são utilizados a mais de 30 anos para automatizar o processo de identificação de produtos, peças e outras coisas (incluindo pessoas). São largamente usados em supermercados, bibliotecas, aeroportos, almoxarifados, transportadoras entre outros. Códigos de barras são sempre compostos de barras verticais, alternando entre barras pretas e brancas, com larguras diferentes. Existem vários padrões de códigos de barras.

Um dos padrões mais simples, porém, menos densos (ocupa mais espaço) é o Código 39. Nesse código, cada caracter (letra ou número) é representado por uma sequência de 9 barras.

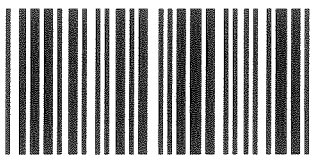
Tabela 1 – Símbolos

Símbolo	Descrição
B	Preta larga
b	Preta fina
W	Branca larga
w	Branca fina

Tabela 2 - Sequências

Caractere	Sequência
*	bWbwBwBwb
0	bwbWBwBwb
1	BwbWbwbwB
2	bwBWbwbwB
3	BwBWbwbwb
4	bwbWBwbwB
5	BwbWBwbwb
6	bwBWBwbwb
7	bwbWbwBwB
8	BwbWbwBwb
9	bwBWbwBwb

Figura 1 – Exemplo de Código 39:



123

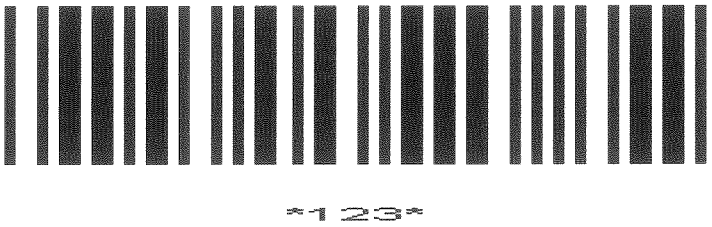
O código sempre inicia e termina com asterisco (*), porém, este é suprimido na exibição do valor após a decodificação. Entre cada caracter, deve haver uma barra branca fina. A Tabela 1 mostra as letras usadas para representar as barras na sequência. A Tabela 2 mostra qual a sequência de cada caracter (nesse problema, consideraremos apenas números, além do asterisco).

Você, um renomado engenheiro de computação, vai ajudar a criar um decodificador de Código 39. Você receberá o que foi lido pelo leitor de código de barras e deverá exibir o número decodificado (sem os asteriscos).

O decodificador fornece as seguintes sequências:

"#"	(sustenido)	– representa uma barra preta larga
"|"	(barra vertical)	– representa uma barra preta fina
" "	(espaço)	– representa uma barra branca larga
""	(sequência vazia)	– representa uma barra branca fina

Segue um exemplo de uma sequência de barras e o seu valor decodificado:

Código de entrada: 

123

Sequência gerada pelo leitor: `|  |##|#|  ||#|#  ||###  ||||  |##|`

Valor decodificado: **123**

Se quebrarmos essa sequência, temos a seguinte associação:

```
|  |##|    -    *
#|  ||#    -    1
|#  ||#    -    2
##  |||    -    3
|  |##|    -    *
```

Note que a saída já é apresentada sem os asteriscos do início e do fim existentes na entrada.

Outra observação importante é que cada caracter é representado sempre por 9 barras (5 pretas e 4 brancas), sendo 3 largas (1 branca e 2 pretas) e 6 finas (3 brancas e 3 pretas). Os caracteres são separados por uma barra fina branca.

IMPORTANTE: O leitor pode ser usado para ler nos 2 sentidos (caso o código seja posicionado para leitura de ponta-cabeça). Portanto, seu programa deve ser capaz de identificar o código em sequências recebidas no sentido inverso (de trás para frente).

Entrada

Sequência de caracteres que representam o código de barras capturado pelo leitor (sentido direto ou inverso) com no no mínimo 12 caracteres e no máximo 72 caracteres. O final dos casos de entrada é representado por um sustenido (#).

Saída

Sequência de números que representam o valor decodificado com no mínimo 0 dígitos e no máximo 10 dígitos.

Exemplos de testes

Entrada	Saída
| |##|#| ||#|# ||### |||| |##|	123
|##| |||| ###|| #|#|| |#|##| |	123
| |##|#| #||#| #||#| #||#| #||| |##|	5555
|##| |||# |#||# |#||# |#||# |#|##| |	5555
| |##||| ##|| |##|	0
|##| ||## |||##| |	0
#	

PROBLEMA G: COLA

Arquivo: `cola.[cpp/c/java]`

Cor: roxo

Limite de tempo: 1s

Descrição do problema

O método expositivo, hoje aplicado na maioria das escolas, faculdades e universidades existe há muito tempo e foi formalizado pela teoria de David Ausubel. O professor Chester, um fã desse método, está tentando criar agora uma técnica para evitar "colas" durante suas provas; para isso ele avaliou as ocorrências de colas e concluiu o seguinte: as "colas" sempre ocorrem entre dois amigos. A partir dessa conclusão ele propôs que as provas fossem realizadas em várias salas simultaneamente e que em cada sala não houvesse nenhum par de alunos que fossem amigos. Prevendo o potencial problema de precisar de muitas salas ele pediu a sua ajuda para, dadas todas as relações de amizade, avaliar se é possível ou não realizar a prova em apenas duas salas.

Entrada

A entrada é constituída de vários casos de teste. Cada caso de teste possui dois números N e A que correspondem à quantidade de alunos que realizarão a prova e a quantidade de relações de amizades entre os alunos, respectivamente. Cada uma das A linhas seguintes contém dois números U e V indicando que U é amigo de V e V é amigo de U. O final de entrada é indicado por N = A = 0.

Restrições

$3 <= N <= 100$
$0 <= A <= N(N-1)/2$
$1 <= U, V <= N$

Saída

Para cada entrada, o programa deverá imprimir sim caso seja possível realizar a prova em apenas duas salas e não caso contrário.

Exemplos de testes

Entrada	Saída
3 3 1 2 2 3 1 3 4 3 1 2 1 3 1 4 5 4 1 2 3 4 4 5 3 5 0 0	nao sim nao

PROBLEMA H: WINTERFELL

Arquivo: `winterfell.[cpp/c/java]`

Cor: amarelo

Limite de tempo: 1s

Descrição do problema

Desde a morte do Rei Robert Baratheon, uma série de fatos foi desencadeada, tornando assim a família Lannister a responsável por governar os Sete Reinos.

Ned Stark, a mão do rei, foi assassinado por JoffreyBaratheon, herdeiro que foi contestado por Ned, após este descobrir que ambos eram filhos de Cersei e seu irmão, Sir Jaime.

A paz que reinava sobre os Sete Reinos foi destruída, com a instabilidade no poder, diversas famílias buscaram seus direitos e diversos exércitos e clãs foram formados.

StannisBaratheon, herdeiro por direito, decidiu armar suas forças para conquistar Porto Real. RobbStark assumiu o trono de Winterfell e busca vingança pela morte de seu pai.

A guerra é uma arte complicada, e Robert busca inspiração no legado deixado por seu pai, para obter sucesso nesta empreitada.

Após convocar um conselho, Robb assumiu o posto de Rei do Norte, descobrindo quem são seus aliados e deixando claro quem são seus inimigos nesta busca por vingança.

O momento é de muita tensão e Robb tem pedido ajuda aos mais experientes, inclusive a você, o novo Mestre de Guerras indicado por Robb.

Com toda sua experiência, você sabe que o próximo passo é identificar os exércitos aliados e inimigos, para poder definir os próximos alvos.

Você então decidiu mandar 10 soldados de extrema confiança para avaliar a região até onde é possível alcançar.

Cada soldado deve retornar com informações sobre os campos de batalha, identificando o número de soldados e as famílias as quais eles pertencem.

Seu dever é utilizar estas informações para informar o Rei do Norte, RobbStark, sobre a situação das tropas no campo de batalha, avaliando se existe uma vantagem numérica ou não.

Entrada

A entrada é composta por dois inteiros ($0 < L <= 100$, $0 < A <= 100$) identicando a dimensão do mapa obtido (Largura x Altura) após a junção das informações trazidas pelos 10 soldados.

A seguir, são recebidos A linhas com L números, indicando a família predominante na área (cada posição do mapa corresponde a 50 soldados).

A família é representada por uma inteiro ($0 < N <= 10$).

A terra vazia é representada pelo valor 0 (zero).

Após receber os mapas, você receberá de RobbStark um inteiro ($0 < N <= 10$) indicando o número de famílias para as quais ele conhece as lealdades, seguido de N linhas, com 1 inteiro indicando a lealdade T da cada família, sendo que 1 representam famílias aliadas e -1 representam famílias cuja posição política é desconhecida.

Os N valores recebidos são referentes a lealdade das famílias em ordem crescente (Família 1, Família 2, Família 3, .., Família N).

O programa encerra com uma entrada L = A = 0.

Saída

A saída é composta por um inteiro M, indicando o número de exércitos encontrados, seguido de M linhas ordenadas de forma crescente para a quantidade de soldados e decrescente para a lealdade, indicando o tamanho do exército e também um inteiro T, indicando se o exército é aliado ou inimigo.

Um exército é definido pela aglomeração de soldados cercados por terras vazias.

Para saber se um exército é aliado ou inimigo, Robb disse que você deve seguir uma regra simples:

- Se existem mais soldados aliados do que inimigos, exército é aliado (1)

- Se existem mais soldados inimigos do que aliados, exército é inimigo (-1)

- Se existem tantos soldados inimigos quanto aliados, deve-se considerar uma batalha, e o exército é neutro (0)

Exemplos de testes

Entrada	Saída
5 4 0 1 3 0 0 0 2 0 0 1 0 0 1 2 4 1 2 0 0 1 4 1 -1 1 -1 10 10 0 0 0 0 0 0 0 0 0 0 1 1 2 0 0 0 1 2 1 0 0 2 2 3 3 0 0 0 1 0 4 4 5 5 5 0 0 0 0 0 0 2 2 3 3 0 0 0 1 0 1 1 2 0 0 0 1 2 1 0 0 0 0 0 0 0 0 4 4 0 0 0 0 0 0 0 0 4 4 0 0 0 0 0 0 0 0 4 4 0 1 1 2 0 0 0 1 2 1 0 5 1 -1 -1 1 1 0 0	3 100 0 150 1 250 1 4 150 1 200 1 650 1 950 -1

PROBLEMA I: ENDOR

Arquivo: `endor.[cpp/c/java]`

Cor: verde claro

Limite de tempo: 1s

Descrição do problema

Um dos grandes marcos da história do universo conhecido foi a Batalha de Endor, em que a segunda Estrela da Morte foi destruída, o imperador derrotado e Darth Vader se redimiu de suas atrocidades.

Você é um cadete da ANI (Academia Novo Império), uma organização extremista que pretende reconstruir os dias gloriosos do Império. No seu teste final para ingressar na tropa imperial, você terá que reviver a histórica batalha e tentar simular diferentes situações para entender onde o Império falhou e evitar novas derrotas.

Sabe-se que uma das decisões mais difíceis antes da Batalha de Endor foi a localização da Estrela da Morte. Seu posicionamento refletiu-se diretamente na energia necessária para a manutenção do escudo na Lua de Endor. Todo cadete sabe desde o primeiro ano que a destruição do escudo ocasionou a derrota do Império.

Pouco antes da Batalha, a Inteligência Imperial havia selecionado diversos pontos ao redor da Lua de Endor onde a Estrela da Morte poderia ter sido posicionada. A escolha final não seguiu nenhum critério, mas acredita-se que se a energia necessária para manter o escudo fosse menor, parte da energia poderia ter sido utilizada em escudos terrestres, frustrando os ataques terrestres rebeldes.

Seu objetivo é justamente escolher a melhor posição que a Estrela da Morte poderia ter ocupado. Passe no teste e sirva ao novo Imperador!

Entrada

A entrada é constituída de vários casos de teste. Cada caso de teste possui a posição da Lua de Endor e um número N (entre 5 e 30) de possíveis posições para a Estrela da Morte. Em seguida, as posições possíveis são informadas. Todas as posições estão informadas com valor relativo ao quadrante de Endor, valores entre zero e 10 mil. A entrada termina quando as coordenadas de Endor são 0, -1, 0.

Saída

Como saída informe a posição em que a Estrela da Morte deveria ser posicionada. Em caso de posições com a mesma eficácia, escolha a primeira que ocorrer.

Exemplos de testes

Entrada	Saída
100 200 300 5 10 20 30 150 250 350 30 20 10 350 250 150 100 200 100 0 0 0 6 10 20 30 150 250 350 30 20 10 350 250 150 100 200 100 5 10 15 0 -1 0	150 250 350 5 10 15

PROBLEMA J: LANÇA-MÍSSEIS

Arquivo: `lanca.[cpp/c/java]`

Cor: marrom

Limite de tempo: 1s

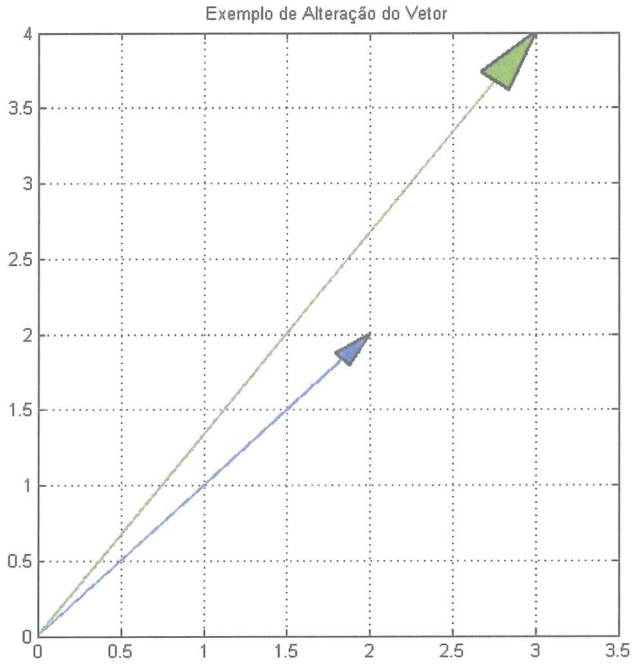
Descrição do problema

Uma fábrica de lança-mísseis está desenvolvendo um novo projeto e contratou a sua empresa para criar um software. O software deverá calcular a direção e a força necessárias para que um desses equipamentos atinja um determinado alvo. O cálculo deverá ser baseado na posição original do equipamento.

Sua empresa decidiu por construir um protótipo usando conceitos de álgebra linear. A direção e a força necessárias para atingir o alvo são representadas, respectivamente, pelo ângulo e pelo módulo de um vetor no plano. Para posicionar e ajustar a força de disparo do lança-mísseis decidiu-se por usar o conceito de transformação linear.

A multiplicação de uma matriz por um vetor causa uma transformação linear no vetor, o que significa que ele pode ter sua direção e seu módulo alterados.

Nesse contexto, sua equipe será responsável por desenvolver o módulo do software que calcule qual deverá ser o deslocamento em graus (sentido anti-horário) e a força de disparo do lança-mísseis. A seguir é apresentado um exemplo:



Entrada

A entrada é composta de vários casos de teste. Cada caso de teste possui seis valores, sendo os dois primeiros as coordenadas x e y da posição original do lança-mísseis, respectivamente. Os próximos dois números compõem a primeira linha da matriz que causará a modificação no vetor. Os últimos dois números compõem a segunda linha. Todas as entradas pertencem ao intervalo [1, 10]. O valor -100 na primeira posição de uma entrada indica o final dos casos de teste.

Saída

Para cada caso de teste deverão ser apresentados o deslocamento em graus do vetor em relação à sua posição original e a alteração do módulo do vetor, nesta ordem. O deslocamento e alteração no módulo devem ser arredondados para o inteiro mais próximo. O ângulo sempre terá como referência o eixo X.

Exemplos de testes

Entrada	Saída
2 2 1 3 4 1 9 7 1 2 7 1 -100	6 10 34 62