Neste trabalho você implementará programas em C++ para resolver os problemas descritos abaixo. Os códigos fonte serão submetidos para correção no sistema Boca (vejam o endereço abaixo), e para cada problema podem ser feitas quantas submissões forem necessárias. No caso de mais de uma submissão correta, apenas a última será avaliada.

Código de Conduta

Este trabalho é **individual.**

Antes de buscar alguma ajuda na resolução do trabalho, considere os seguintes pontos.

1. Se alguém te explicar a lógica a ser implementada antes de você tentar desenvolver uma, você perderá a oportunidade de exercitar esta etapa do processo de solução de problemas.
2. Se alguém te mostrar uma solução em código C ou C++ antes de você tentar desenvolver ou encontrar erros na sua implementação, você perderá a oportunidade de exercitar suas habilidades em programação.

Quando terminar um código, jamais passe esse o código para um colega. Ajude-o explicando a lógica da solução ou encontrando erros na implementação - jamais entregue o seu código para outros alunos. Trabalhos com códigos idênticos ou muito parecidos serão anulados.

Comentários e indentação

Códigos que não estejam devidamente indentados podem receber desconto na nota. Além disso, devem existir comentários explicativos nas partes mais complexas da solução.

Link para o envio dos programas no sistema Boca

As soluções devem ser enviadas através do seguinte sítio:

http://boca.dpi.ufv.br/trabalhos/

O seu login e senha são iguais a sua matrícula. Você deverá trocar a sua senha no primeiro acesso ao sistema. Você poderá receber alguma penalização na avaliação do trabalho caso não troque sua senha.

Para trocar a senha clique em *Options* no menu do sistema.

A – Gasto de Gasolina

gasolina.c ou gasolina.cpp

João está interessado em estimar o gasto de gasolina de suas viagens. Para tal, ele necessita de um programa para fazer essas estimativas considerando o gasto médio de gasolina do carro, o tempo de duração da viagem e a velocidade média na viagem.

Entrada

A entrada terá três inteiros, o primeiro indicando o tempo gasto na viagem (horas), o segundo indicando a velocidade média da viagem (km/h) e o último indicando o consumo médio do carro (km/l).

Saída

A saída deverá indicar a quantidade, em litros, de gasolina consumida na viagem. A quantidade em litros deve ser exibida com três casas decimais.

Exemplos

Entrada	Saída
12 78 5	187.200

Entrada	Saída
29 104 9	335.111

Entrada	Saída
13 87 13	87.000

B – Jogo da Velha

velha.c ou velha.cpp

João está cansado de ter que verificar mentalmente o resultado de partidas do jogo da velha. Para ajudar, crie um programa em C++ que recebe como entrada uma sequência de jogadas do jogo da velha e retorne o vencedor da partida.

Entrada

A entrada será uma sequência de pares de inteiros (i, j) com 1 ≤ i,j ≤ 3, onde cada par indica a jogada de um jogador (jogador 1 ou jogador 0). Na nossa versão do jogo da velha o jogador 1 sempre iniciará as partidas, seguido do jogador 0; as jogadas serão alternadas entre os jogadores até o término da partida. Se jogador 1 entrar com os valores (i, j), marcaremos com o valor 1 a posição (i, j) da matriz de 3 linhas e 3 colunas que define o jogo; vejam a figura abaixo.

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

Uma partida do jogo da velha termina quando todas as 9 entradas da matriz são preenchidas ou quando um jogador completa uma linha, coluna ou diagonal da matriz.

A sequência de pares (i, j) mostrada no quadro abaixo resulta na configuração do jogo da velha mostrada à direita no quadro. Esse jogo termina com a vitória do jogador 0 pois o jogador completou uma sequência de zeros na diagonal secundária da matriz.

```
1 1
1 3
2 3
2 1
3 2
2 2
3 3
3 1
```

1		0
0	0	1
0	1	1

Saída

A saída deve indicar se o vencedor da partida foi o jogador 1, jogador 0, ou se ocorreu um empate.

Exemplos

Entrada	Saída
2 2 2 3 1 2 1 1 3 2	Vitoria 1

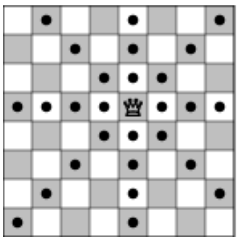
Entrada	Saída
3 2 2 1 3 3 2 2 1 1 2 3	Vitoria 0

Entrada	Saída
1 3 1 2 3 2 3 1 2 1 2 3 3 3 1 1 2 2	Empate

C – Movimentos da Rainha

rainha.c ou rainha.cpp

A rainha é uma das peças mais importantes de um jogo de xadrez. A sua importância se deve a sua flexibilidade, pois ela pode se mover um número arbitrário de casas nas 8 direções cardeais, como ilustrado na figura abaixo.



Neste problema você irá escrever um programa para calcular o número de movimentos mínimos que a rainha necessita fazer para sair de uma posição de origem e chegar em uma posição de destino. As linhas e colunas de um jogo de xadrez são numeradas de 1 a 8, as linhas de cima para baixo e as colunas da esquerda para direita.

Entrada

A entrada terá quatro inteiros **X1**, **Y1**, **X2**, **Y2**, onde o par (**X1**, **Y1**) representa a posição de origem da rainha e o par (**X2**, **Y2**) a posição de destino. Tem-se que 1 ≤ **X1**, **Y1**, **X2**, **Y2** ≤ 8.

Saída

A saída deverá indicar o número mínimo de movimentos da rainha para sair de (**X1**, **Y1**) e chegar em (**X2**, **Y2**).

Exemplos

Entrada	Saída
3 6 8 1	1

Entrada	Saída
5 5 3 6	2

D – Distância a Percorrer

distancia.c ou distancia.cpp

Os movimentos nas oito direções cardeais da rainha, como explicados no problema anterior, são utilizados em algoritmos de Inteligência Artificial (IA) implementados em jogos de computador como *Baldur's Gate*, *Neverwinter Nights* e *Dragon Age*. O sistema de IA desses jogos é responsável por encontrar um caminho entre um ponto inicial e um ponto destino em um mapa. Um componente importante desses algoritmos é uma função que calcula a distância mínima a ser percorrida entre a posição inicial e a de destino. Você irá implementar um programa de computador que calcula a distância mínima de um caminho entre a posição inicial e a posição destino.

Assim como no jogo de Xadrez, o mapa é representado por um tabuleiro de 8 linhas e 8 colunas, onde as linhas são numeradas de 1 a 8 de cima para baixo e as colunas também de 1 a 8 da esquerda para direita.

Assim como a rainha, os personagens do nosso jogo também se movem nas 8 direções cardeais. No entanto, os personagens se movimentam uma posição por vez. Além disso, movimentos distintos possuem custos distintos (distâncias percorridas distintas). Isto é, os movimentos na diagonal custam 1,5 e os movimentos na horizontal e vertical custam 1,0.

Por exemplo, a distância mínima entre a posição (1, 8) e (8, 1) no mapa é igual a 10,5 (o equivalente a 7 movimentos diagonais).

Entrada

A entrada terá quatro inteiros **X1**, **Y1**, **X2**, **Y2**, onde o par (**X1**, **Y1**) representa a posição de origem do personagem e o par (**X2**, **Y2**) a posição de destino. Tem-se que $1 \leq$ **X1**, **Y1**, **X2**, **Y2** ≤ 8.

Saída

A saída deverá indicar a distância mínima a ser percorrida pelo personagem entre a origem (**X1**, **Y1**) e destino (**X2**, **Y2**). A distância deverá ser exibida com uma casa decimal.

Exemplos

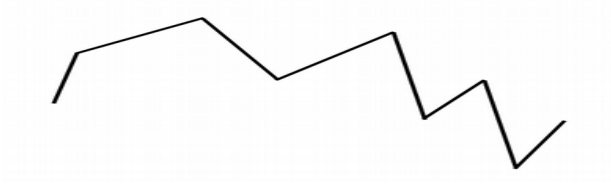
Entrada	Saída
4 4 5 2	2.5

E – PETR4

petr4.c ou petr4.cpp

Devido ao cenário político-econômico atual, o preço das ações que formam o índice BOVESPA (iBOVESPA) tem sofrido fortes oscilações. Uma das principais ações do iBOVESPA é a PETR4 – ações preferenciais da Petrobras. Para estimar quando o preço da PETR4 irá se estabilizar em uma tendência definida, um corretor necessita calcular o número de *topos* e *fundos* da série histórica da PETR4.

Um topo é caracterizado pela mudança de preço de tendência de alta para baixa; um fundo é definido de maneira análoga, quando há mudança de preço de baixa para alta. Por exemplo, a figura abaixo mostra o preço de fechamento da PETR4 de nove dias seguidos de pregão; a série contém 3 topos e 3 fundos. O preço do primeiro e último dias da série não são considerados nem como topo e nem como fundo.



Entrada

A entrada terá um inteiro **N** > 2 indicando o número de dias da série e uma sequência de números **P1**, **P2**, ..., **PN**, onde para quaisquer **Pi** e **Pj** adjacentes tem-se que **Pi ≠ Pj** e **Pi** > 0 .

Saída

A saída deverá conter o número de topos e fundos separados por um espaço.

Exemplos

Entrada	Saída
5 5.94 4.31 7.51 8.8 5.81	1 1

Entrada	Saída
7 2.02 2.07 6.38 3.34 2.88 2.01 8.02	1 1

F – Estações do ano

estacao.c ou estacao.cpp

A data exata de início de cada estação é definida pelos solstícios e equinócios e pode variar de ano pra ano. Vamos assumir as seguintes datas de início das estações:

Outono	Inverno	Primavera	Verão
20 de março	21 de junho	22 de setembro	21 de dezembro

Sua tarefa é fazer um programa que lê uma data (dia e mês) e escreve a estação correta do ano, de acordo com a tabela acima.

Entrada

O programa terá um caso de teste que é descrito em uma linha contendo dois inteiros, **D** e **M**, representando o dia e o mês. Restrição: **D** e **M** representam uma data válida.

Saída

Seu programa deve gerar duas linhas na saída: a primeira contém a data por extenso, no formato "**D** `de` <mês>", sendo **D** o dia fornecido na entrada e <mês> o nome do mês **M**, com inicial minúscula; a segunda linha deve conter uma das palavras, "`Primavera`", "`Verao`", "`Outono`" ou "`Inverno`", dependendo da estação da data da entrada. Não use cedilhas nem acentos na saída. Veja os exemplos.

Exemplos

Entrada	Saída
`4 4`	`4 de abril` `Outono`

Entrada	Saída
`25 12`	`25 de dezembro` `Verao`

Entrada	Saída
`20 3`	`20 de marco` `Outono`

G – Salto de Esqui

esqui.c ou esqui.cpp

Nesta questão você deve fazer um programa para calcular a pontuação de um competidor de "Salto de esqui". Nesse esporte o atleta desce de esqui uma rampa e salta da rampa tentando aterrissar o mais distante possível. A pontuação leva em conta a distância e o estilo do salto.

Para a distância, existe uma marca alvo, chamada de ponto *K*. Se o atleta aterrissa na marca, ganha 60 pontos. Se não aterrissa na marca ganha ou perde 1.8 pontos por cada metro além ou aquém da marca. Assim, em uma competição em que a marca está a 120 metros (k-120), um atleta que aterrissa na marca dos 120 metros ganha 60 pontos. Outro que aterrissa a 122 metros ganha 63.6 pontos (60 + 3.6 pontos pelos 2 metros depois da marca), e um que aterrissa a 117 metros ganha 54.6 pontos (60 – 5.4 pontos pelos 3 metros antes da marca).

Para o estilo, existem 5 juízes que avaliam a posição do esqui durante o salto, o equilíbrio e postura do corpo do atleta e a aterrissagem. Cada juiz dá uma nota de valor máximo 20. A nota do estilo é a soma das notas atribuídas pelo juízes, desconsiderando-se a menor e a maior das notas. Assim, um atleta que recebeu dos juízes as notas 19 19 18 18.5 e 19 terá nota de estilo 56.5, pois serão descartadas a nota 18 (menor) e uma das notas 19 (maior).

A pontuação do atleta é a soma da nota da distância com a nota de estilo.

Você deve fazer um programa que lê os dados do salto – distância e notas dos juízes, calcula e escreve a pontuação do atleta considerando uma competição K-120.

Entrada

O programa terá apenas um caso de teste que é descrito em duas linhas: a primeira contém um valor real **D**, que indica a distância do salto, em metros; a segunda contém cinco valores reais, **J1 J2 J3 J4 J5**, que são as notas dos juízes. Restrição: 0 ≤ **D** ≤ 250 e 0 ≤ **J1 J2 J3 J4 J5** ≤ 20.

Saída

Seu programa gera apenas uma linha de saída, contendo um valor real com uma casa decimal representando a pontuação do atleta. Não se esqueça de pular uma linha após a impressão do valor.

Exemplos

Entrada	Saída
120 17 17 18 17 17	111.0

Entrada	Saída
135 18 18.5 18.5 19 19	143.0

Entrada	Saída
111 18 17 17 16.5 17	94.8

H – Máquina automática

maquina.c ou maquina.cpp

O DCE quer instalar uma máquina automática de venda de chocolates, salgadinhos, refrigerante, etc. Você foi convidado a programar o software que contabiliza o valor inserido pelo comprador e diz se ele é suficiente para comprar o produto escolhido.

Nessa primeira versão os valores estarão todos em centavos. Por exemplo, o valor de um produto que custa R$2,50 será representado por 250. E uma moeda de R$1,00 terá seu valor representado por 100.

Entrada

O programa terá apenas um caso de teste. A primeira linha do caso de teste contém um valor inteiro **P**, que é o preço do produto escolhido. Em seguida virá uma linha contendo uma lista de valores **V**, cada um representando o valor de uma moeda inserida na máquina. A lista termina quando aparecer um valor 0, indicando que o comprador apertou o botão "liberar produto e troco".

Restrição: 1 ≤ **P** ≤ 5000, **V** assumirá um dos seguintes valores: {1, 5, 10, 25, 50, 100}.

Saída

A saída de seu programa pode ser de dois tipos:

- se o comprador não inseriu dinheiro suficiente para comprar o produto, escreva "`Saldo insuficiente.`".

- caso contrário escreva "`Troco de` **X** `centavos.`", sendo **X** o valor apropriado.

Não se esqueça de pular uma linha após a impressão da saída.

Exemplos

Entrada	Saída
100 50 25 10 10 10 0	Troco de 5 centavos.

Entrada	Saída
105 50 25 10 10 10 0	Troco de 0 centavos.

Entrada	Saída
110 50 25 10 10 10 0	Saldo insuficiente.

Entrada	Saída
263 50 50 100 100 0	Troco de 37 centavos.

Entrada	Saída
199 25 5 10 100 100 0	Troco de 41 centavos.

I – Máquina plus

maquinap.c ou maquinap.cpp

Agora que seu software para cálculo do valor pago pelo comprador foi testado e funciona corretamente, ele deve ser modificado para também aceitar cédulas e não apenas moedas.

A cada valor inserido pelo comprador o software receberá uma letra e um valor inteiro. A letra será '`C`' ou '`M`', identificando se foi inserida uma cédula ou uma moeda. Caso seja uma cédula o valor indica o valor da cédula, em reais. Caso seja uma moeda indica o valor da moeda, em centavos. Assim, `C 1` representa a inserção de uma cédula de 1 real, enquanto `M 1` representa a inserção de uma moeda de 1 centavo. Note que a moeda de 1 real é representada por `M 100`.

O objetivo é o mesmo da questão anterior: contabilizar o valor total inserido e informar o troco, se houver. Além da possibilidade de inserir moedas, o software deve considerar os produtos e valores pré-definidos da lista a seguir.

Número	Descrição do produto	Preço
1	Batata	R$ 4,30
2	Suco	R$ 2,70
3	Guaraná	R$ 1,43

Entrada

O programa terá apenas um caso de teste. A primeira linha do caso de teste contém um valor inteiro **P**, que é o número do produto escolhido. Em seguida virá uma sequência de linhas, cada uma contendo um caractere **T** e um valor **V**, representando respectivamente se foi inserida uma cédula ou moeda e o valor dela. A lista termina quando aparecer um valor `0`, seja pra cédula ou moeda, indicando que o comprador apertou o botão "liberar produto e troco".

Restrições: 1 ≤ **P** ≤ 10, **T** assumirá um dos dois valores: {'`C`', '`M`'}, **V** um valor válido de cédula ou moeda brasileira.

Saída

A saída de seu programa pode ser de três tipos:

- se o comprador escolheu um produto diferente de 1, 2 e 3, escreva a mensagem "`Produto inexistente.`"

- se o comprador não inseriu dinheiro suficiente para comprar o produto, escreva "`Saldo insuficiente.`"

- caso contrário escreva "`Troco de` **X** `centavos.`", sendo **X** o valor apropriado.

Não se esqueça de pular uma linha após a impressão da saída.

Exemplos

Entrada	Saída
1 C 5 C 0	Troco de 70 centavos.

Entrada	Saída
2 C 2 M 50 M 25 M 0	Troco de 5 centavos.

Entrada	Saída
4 C 100 C 20 C 0	Produto inexistente.

Entrada	Saída
3 C 10 C 0	Troco de 857 centavos.

J – Máquina plus plus!

maquinapp.c ou maquinapp.cpp

Como última funcionalidade do software antes de instalar a máquina no DCE, você deve implementar a parte que "devolve" o troco. Nesse caso instruções que serão enviadas para o mecanismo da máquina liberar cédulas e moedas.

Da mesma forma que no problema anterior, o consumidor escolhe um produto e insere cédulas e moedas. Após verificar se o produto existe e se foi inserido um total suficiente, deve-se imprimir as instruções de troco, que são uma sequência de linhas contendo um caractere '`C`' ou '`M`', identificando se deve ser devolvida uma cédula ou moeda, e o valor (da cédula ou moeda). Os valores válidos são:

`C 100, C 50, C 20, C 10, C 5, C 2,`

`M 100, M 50, M 25, M 10, M 5, M 1.`

Note que a máquina não devolve cédulas de R$ 1, pois já saíram de circulação, mas devolve moedas de R$ 1 (representadas por `M 100`).

Para minimizar o número de cédulas e moedas devolvidas a máquina devolve os valores na ordem acima (do maior ao menor valor). Ou seja, um troco de 30 centavos será devolvido como `M 25 M 5`, e não como `M 10 M 10 M 10` e nem como `M 5 M 25`.

Entrada

A entrada segue exatamente o mesmo formato da questão anterior.

Saída

A saída de seu programa pode ser de três tipos:

- se o comprador escolheu um produto diferente de 1, 2 e 3, escreva a mensagem "`Produto inexistente.`"

- se o comprador não inseriu dinheiro suficiente para comprar o produto, escreva "`Saldo insuficiente.`".

- caso contrário o programa deve simular a devolução do troco imprimindo o valor das cédulas e moedas devolvidas, uma em cada linha, na ordem especificada acima.

Não se esqueça de pular uma linha após a impressão da saída.

Exemplos

Entrada	Saída
1 C 5 C 0	M 50 M 10 M 10

Entrada	Saída
2 C 2 M 50 M 25 M 0	M 5

Entrada	Saída
4 C 100 C 20 C 0	Produto inexistente.

Entrada	Saída
3 C 50 C 0	C 20 C 20 C 5 C 2 M 100 M 50 M 5 M 1 M 1

Obs.: para evitar problemas de precisão use somente números inteiros, ou seja, internamente represente tudo em centavos; por exemplo C 20 são 2000 centavos.