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 contar 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 receberão nota **ZERO**.

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 – Compactador

compacta.c ou compacta.cpp

Os compactadores de arquivos trabalham retirando redundância do arquivo. Por exemplo, palavras ou até partes de um texto que aparecem várias vezes podem ser escritas uma vez só, e para as demais ocorrências são gravadas apenas referências a algo que já apareceu antes. Em imagens, grandes blocos de cores iguais podem ser resumidos a uma cor e quantas vezes ela é repetida.

Um compactador bastante simples é um que troca uma sequência de caracteres iguais por um par de valores: quantas vezes o caractere ocorre, e qual é esse caractere. Veja o exemplo a seguir:

Arquivo original: `GGGGTTTTTCCAAAAAAAAAAAGGGGAAAAAACTTTTTAAAGGGGGGGGGGGGG`
Compactado: `4G5T2C11A4G6A1C5T3A13G`

A descompactação também é simples, lê-se um número e um caractere, e esse caractere é escrito repetidamente tal número de vezes.

Note que esse compactador só reduz realmente o tamanho do arquivo se houver sequência com mais de 2 caracteres iguais consecutivos. De fato, quando há apenas 2 caracteres não há ganho algum (como na primeira sequência de `C`'s, onde `CC` é substituído por `2C`). E quando não há repetição, a compactação na verdade aumenta o tamanho (como na segunda sequência de `C`, onde `C` é substituído por `1C`).

Você deve fazer um programa para fazer a compactação de um arquivo da forma como mostrado acima. O arquivo contém apenas letras (maiúsculas ou minúsculas) e possui um *flag* no final para sinalizar o final da entrada (um caractere '`.`'). O "arquivo" na verdade será uma sequência de letras lidas como entrada do programa.

Entrada

O programa terá apenas um caso de teste.

O caso de teste é descrito em uma única linha, que contém uma sequência de letras terminada por um caractere '`.`'. Não há limite para a quantidade total de letras mas nenhuma sequência tem mais de 1000 letras iguais seguidas.

Saída

Seu programa gera apenas uma linha de saída, contendo o resultado da compactação da entrada. Imprima um '`0`' (zero) após o final da sequência compactada.

Não se esqueça de encerrar a linha após a impressão do resultado.

Exemplos

Entrada	Saída
AAAAAAAAAARRRRAAAAA.	10A4R5A0

Entrada	Saída
AAAAaaaAAAAAAa.	4A3a6A1a0

Entrada	Saída
GOOOOOOOOOOOOOOLLLLLL.	1G15O6L0

Entrada	Saída
Pernambuco.	1P1e1r1n1a1m1b1u1c1o0

B – Descompactador

descompacta.c ou descompacta.cpp

No problema anterior você teve que fazer um compactador de arquivos. Agora você deve fazer um descompactador, um programa que volta o arquivo compactado ao seu conteúdo original! Conforme descrição no problema anterior, o arquivo compactado contém uma sequência de pares de valores. Cada par é formado por um número indicando a quantidade e um caractere que se repete esse número de vezes.

Observação: consideramos que o arquivo original contém apenas letras; se existissem números haveria ambiguidade na descompactação (por exemplo não daria pra dizer se 31415 deveria ser descompactado como 31 '4's seguidos de 1 '5', ou 3 '1's seguidos de 41 '5's).

Entrada

O programa terá apenas um caso de teste.

O caso de teste é descrito em uma única linha e contém uma sequência de caracteres que representa um arquivo compactado válido, ou seja, uma sequência de números e letras alternados. O final da entrada é indicado por um '0' (zero).

Saída

Seu programa gera apenas uma linha de saída, que é o arquivo descompactado conforme as regras descritas, com um '.' no final.Não se esqueça de encerrar a linha após a impressão do valor.

Exemplos

Entrada	Saída
10A4R5A0	AAAAAAAAAARRRRAAAAA.

Entrada	Saída
4A3a6A1a0	AAAAaaaAAAAAAa.

Entrada	Saída
1G15O6L0	GOOOOOOOOOOOOOOLLLLLL.

Entrada	Saída
1P1e1r1n1a1m1b1u1c1o0	Pernambuco.

C – Dígitos Adornados

adornados.c ou adornados.cpp

A empresa Dígitos Adornados é concorrente da Sete Especial, empresa que produz números para casas e apartamentos onde o número 7 é mais adornado que os demais. A principal vantagem que a Dígitos Adornados possui sobre as concorrentes é que seus números são todos igualmente adornados e caros.

Ao saber que você implementou para a Sete Especial um programa de computador que calcula a quantidade de números 7 em um dado intervalo, os diretores da Dígitos Adornados estão interessados em te contratar para escrever um programa um pouco mais elaborado. O seu programa irá receber como entrada dois inteiros *a* e *b*, e irá retornar o número de vezes que cada um dos 10 diferentes dígitos aparecem no intervalo *[a, b]*.

Entrada

O programa terá apenas um caso de teste contendo dois inteiros *a* e *b*. Pode-se assumir que *0 < a, b < 100000*

Saída

Seu programa gera 10 números, um para cada dígito, mostrando o número de vezes que cada dígito aparece no intervalo *[a, b]*. Isto é, o primeiro número impresso indicará a quantidade de 0s que aparecem no intervalo *[a, b]*, o segundo número indicará a quantidade de 1s que aparecem no intervalo *[a, b]*, e assim por diante. Lembre-se de quebrar a linha ao imprimir o último valor.

Exemplos

Entrada	Saída
1 10	1 2 1 1 1 1 1 1 1 1

Entrada	Saída
1 155	25 92 36 36 36 32 25 25 25 25

D – Suinocultura

suinos.c ou suinos.cpp

Alunos do curso de Zootecnia da Universidade Federal de Viçosa estão terminando um experimento em que foi testada uma dieta rica em Cr (cromo) em leitões. Todos os *N* animais que participaram no experimento serão pesados e os alunos deverão calcular o peso médio, o desvio padrão, e a quantidade de animais com peso acima da média.

Devido a vários fatores que podem influenciar os experimentos dessa natureza, é preciso desconsiderar *outliers*. Isto é, ao calcular as estatísticas especificadas acima, precisam-se desconsiderar os pesos dos dois animais mais pesados e os pesos dos dois animais mais leves. Pode-se assumir que todos os valores de peso serão distintos.

Seja P o conjunto de valores de peso dos animais após removermos os *outliers*. A média e o desvio padrão dos valores de P deverão ser calculados da seguinte forma:

$$\mu = \frac{1}{N-4}\sum_{p \in P} p \qquad \sqrt{\frac{1}{N-4}\sum_{p \in P}(p-\mu)^2}$$

Entrada

O programa terá apenas um caso de teste. O programa irá receber um valor inteiro $10 < N < 100$ e uma sequência de *N* de valores reais *p* indicando os pesos dos animais. Pode-se assumir que $60.00 < p < 90.00$.

Saída

O programa irá imprimir a média, desvio padrão e a quantidade de valores acima da média para o conjunto P. O valores serão impressos em uma mesma linha separados por espaço e com duas casas decimais de precisão para a média e desvio padrão. Lembre-se de quebrar a linha ao imprimir a quantidade de animais acima da média.

Exemplo

Entrada	Saída
11 61 62 63 64 65 66 67 68 69 70 71	66.00 2.00 3

E - Hamming

hamming.c ou hamming.cpp

O *peso de Hamming* para uma cadeia de caracteres é o número de símbolos diferentes do símbolo nulo do alfabeto utilizado. Este número possui aplicações em criptografia, teoria da informação, xadrez jogado por computador, entre outras áreas. No caso de números inteiros, o *peso de Hamming* é dado pela quantidade de dígitos diferentes de zero. Em particular, para uma representação binária, o peso é dado pela soma dos dígitos. Faça um programa para calcular, **de forma recursiva**, o *peso de Hamming* para um dado número binário. Utilize o seguinte protótipo para a função recursiva:

```
int hamming(long x);
```

Entrada

A entrada é composta por um número binário (inteiro) x.
Restrição: 0 ≤ x ≤ 1000000000.

Saída

Seu programa deve informar o valor (inteiro) do *peso de Hamming* para uma dada entrada binária.

Exemplos

Entrada	Saída
1000000	1

Entrada	Saída
1110010	4

Entrada	Saída
111111111	9