GSI064 - RESOLUÇÃO DE PROBLEMAS

14 de Maio de 2022

MINIMARATONA DE STRINGS

CADERNO DE PROBLEMAS

Problema A. Poodle

Nome do arquivo fonte: `poodle.c`, `poodle.cpp`, ou `poodle.java`

Maria vive perdendo coisas dentro de casa. Desde pequena, tudo em que ela põe a mão desaparece. Isso acontece porque ela é muito desorganizada, deixa tudo espalhado pela casa, tornando humanamente impossível localizar algum objeto no meio de tanta confusão. Ela sempre contou com a ajuda infalível de seu cãozinho Poodle, que consegue localizar seus objetos perdidos. Uma vez ela queria me mostrar a eficiência do seu Poodle. Escondeu propositalmente uma bola em um dos quartos, e gritou: "Poooooooodle!". Então ela disse "Bola" e ele partiu para buscá-la. Ela ficou preocupada porque depois de 30 segundos ele ainda não tinha retornado com a bola. A surpresa foi que logo depois ele apareceu, triufante, carregando 4 bolas!!! A que Maria acabara de esconder e outras 3 de seus filhos, que ela nem se lembrava que existiam, e muito menos onde estavam!

Atualmente Maria faz pós-graduação em Computação. Seu projeto final é uma ferramenta de busca. Em homenagem a seu cãozinho que sempre buscou suas coisas, ela batizou seu projeto de Poodle. A ideia é simples: dada uma palavra, Poodle faz uma busca no disco e retorna todos os documentos que contém a dada palavra. Como no caso real da bola que comentei acima, na maioria das vezes a busca retorna bem mais resultados que o esperado. Os resultados são então exibidos agrupados em páginas. Por exemplo, se a ferramenta for configurada para exibir 10 resultados por página, e a busca retornar 143 resultados, eles serão exibidos em 15 páginas: 14 delas com 10 em cada uma, e a última com os 3 restantes.

A ferramenta já está pronta, e funciona muito bem. Mas Maria teve a feliz ideia de enfeitar o trabalho na tentativa de ganhar mais pontos... ao exibir o resultado de uma busca, Poodle mostra um logotipo com o nome da ferramenta, sendo que os o's de Poodle podem ser dois ou mais, dependendo da quantidade de páginas de resultado. A ideia é que "Poodle" seja escrito com tantas letras quantas forem as páginas de resultado, repetindo o's quando necessário. No exemplo acima, o logotipo seria "Pooooooooooodle", que contém 15 letras.

Naturalmente, se a quantidade de páginas de resultados for inferior a 6, a palavra Poodle não será cortada, o logotipo será Poodle. E, para evitar que o logotipo fique tão grande que nem caiba na tela de resultados, ele será limitado a um máximo de 20 letras, mesmo que a quantidade de páginas de resultado seja superior a 20.

Sua tarefa é ajudar Maria a montar o logotipo.

Entrada

Há vários casos de teste.

Cada caso de teste é uma linha contendo dois números inteiros, N e P, sendo N o número de documentos encontrados pelo Poodle, e P o número de resultados exibidos por página ($1 \leq N \leq 1.000.000$, $1 \leq P \leq 100$). A entrada termina quando $N = P = 0$.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma linha na saída contendo a palavra Poodle, ajustando a quantidade de o's de acordo com as regras descritas no enunciado.

Exemplos

Entrada	Saída
20 4 143 10 42 5 80 3 0 0	Poodle Pooooooooooodle Pooooodle Poooooooooooooooodle

Problema B

Cores

Arquivo fonte: cores.c, cores.cpp ou cores.java

Faça um programa que determine se uma cor é uma cor primária, secundária ou outra. Uma cor é considerada:

- Primária, se for, **amarela**, **vermelha** ou **azul**;
- Secundária se for **laranja**, **verde** ou **roxa**;
- Outra, caso não seja nenhuma das opções anteriores.

Entrada

Uma cor, conforme descrito acima.

Saída

Se está cor é uma cor “primaria”, “secundaria”, ou “outra”.

Exemplo de Entrada (Caso de teste 1)	**Exemplo de saída (Caso de teste 1)**
amarela	primaria

Exemplo de Entrada (Caso de teste 2)	**Exemplo de saída (Caso de teste 2)**
marrom	outra

Problema C

Escada

Arquivo fonte: escada.c, escada.cpp, escada.java

A empresa de criação de jogos “Tudo Muito Louco” fundada por Alex Pipe precisa de seus serviços para automatizar uma rotina de um jogo que estão criando. De acordo com a nível N em que o jogador estiver sua rotina deve criar uma escada com N níveis.

Entrada

Um numeral N que representa o nível do jogador, $0 < N \leq 100$.

Saída

Uma escada formada por espaçamentos e pelo caractere “#” conforme exemplo abaixo:

Exemplo de entrada (Caso de teste 1):	Exemplo de saída (Caso de teste 1):
1	#

Exemplo de entrada (Caso de teste 2):	Exemplo de saída (Caso de teste 2):
5	# ## ### #### #####

Problema d. Palavras Ordenadas

Arquivo-fonte: `palavras.c`, `palavras.cpp` ou `palavras.java`

Palavras como "belo", "fino", "cruz" e "flor" possuem a interessante propriedade de que suas letras aparecem todas em ordem alfabética. Por exemplo, em "flor", `f` vem antes de `l` no alfabeto, `l` vem antes de `o`, e `o` vem antes de `r`. Dizemos que palavras com essa propriedade são *ordenadas*.

Dada uma palavra, determine se ela é ordenada.

Entrada

A entrada começa com uma linha contendo apenas um inteiro P, que representa o número de palavras que serão dadas na entrada. Em seguida, há P linhas, cada uma contendo uma palavra.

Cada palavra é composta apenas por letras minúsculas ou maiúsculas, sem acentos ou pontuação. Você pode supor que não há palavras com mais de 42 caracteres.

Saída

Para cada palavra dada na entrada, imprima uma linha na saída no formato `<palavra>: <r>`, onde `<palavra>` é a palavra em si, e `<r>` é `O` se a palavra for ordenada e `N` caso contrário.

Exemplos

Entrada	Saída
8 a belo fiNo Cruz flor batata abracadabra aaaabc	a: O belo: O fiNo: O Cruz: O flor: O batata: N abracadabra: N aaaabc: N

Note que `aaaabc` *não* é uma palavra ordenada, porque `a` não vem antes de `a` no alfabeto.

Problema E. Alienígenas!

Arquivo-fonte: `alienigenas.c`, `alienigenas.cpp` ou `alienigenas.java`

A NASA finalmente anunciou a descoberta de vida fora da Terra. Uma sonda enviada à Marte detectou diversas formas de vida, e coletou indivíduos, que foram trazidos de volta a nosso planeta.

Internamente, esses indivíduos alienígenas são bem diferentes das formas de vida terrestres. Enquanto que, por aqui, a *ordem* das bases no DNA determina as características do indivíduo, nos marcianos só importa quantas cópias de cada base estão presentes.

Mais especificamente, o "mDNA" (ou DNA marciano) é composto de 15 bases, representadas pelas letras de A a O. Uma espécie é unicamente determinada pela quantidade de cada uma dessas bases que aparece em seu mDNA. Dois indivíduos de uma mesma espécie podem ter sequências de mDNA distintas. Por exemplo, um indivíduo com mDNA `AABABJJ` é da mesma espécie que um indivíduo com mDNA `AJBAJBA`, pois ambos possuem três bases A, duas bases B e duas bases J. Indivíduos `ABACA` e `ABABCA`, porém são de espécies diferentes: enquanto o primeiro possui apenas uma base B, o segundo possui duas cópias dessa base.

São dadas as sequências de mDNA de n indivíduos. Determine quantas espécies distintas estão presentes.

Entrada

A entrada possui vários casos de teste. Cada caso de teste começa com um inteiro n, que representa o número de indivíduos ($0 < n < 65536$). Em seguida há n linhas, cada uma das quais com a descrição do mDNA de um indivíduo. Essa descrição é composta apenas por letras maiúsculas entre A e O (inclusive), e possui no máximo 100 caracteres.

A entrada termina com $n = 0$, que não deve ser processado.

Saída

Para cada caso de teste, imprima uma linha na saída contendo um único inteiro e, o número de espécies distintas presentes entre os indivíduos do caso de teste.

Exemplos

Entrada	Saída
3 AAA AAB ABC 4 AAAB AABA ABAA BAAA 0	3 1

Problema F
Codifica

Arquivo fonte: codifica.c, codifica.cpp ou codifica.java

Foi criada uma nova técnica para codificar mensagens a partir da inserção de caracteres, randomicamente, em palavras enviadas. Para validar este método é necessário escrever um programa que verifica se a mensagem está realmente codificada.

Desta forma, dadas duas palavras ***s*** e ***t***, é preciso decidir se ***s*** é uma subsequência de ***t***. Por exemplo, se podem ser removidos caracteres de ***t*** tal que a concatenação dos caracteres restantes seja ***s***.

Entrada

A entrada contém vários casos de teste, e cada linha possui duas palavras ***s*** e ***t*** formadas por caracteres alphanuméricos ASCII, separadas por espaço. As palavras terão até 1000 caracteres.
A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada produzir uma linha na saída com “sim” ou “nao” respectivas à informação se ***s*** é uma subsequência de ***t***, ou não.
A saída deve ser escrita na saída padrão.

Exemplo de entrada

```
sequencia subsequencia
person compression
Mbuni Muitobomeumnovohorizonte
VERDI vivaVittorioEmanueleReDiItalia
PLANETA Planeta
```

Saída para o exemplo de entrada

```
sim
nao
sim
sim
nao
```

Problema G

Palíndromo

Arquivo fonte: palindromo.c, palindromo.cpp, palindromo.java

Certamente você conhece a frase: “Socorram-me subi no ônibus em Marrocos”. Essa frase é bastante conhecida, devido ao fato dessa frase ser um palíndromo, isto é, ela pode ser lida da direita para a esquerda ou da esquerda para a direita que a informação transmitida é a mesma.

Bem como frases, existem palavras que também são palíndromos, como por exemplo, “arara” ou “ovo”.

Capicua é um número, ou conjunto de números, que são palíndromos, por exemplo 131, ou 424.

Neste problema você terá que responder a seguinte pergunta: Quais são o maior e menor números palíndromos dentro do intervalo fornecido? Caso não haja nenhum número palíndromo no intervalo seu programa deve imprimir uma informação na tela.

Entrada

Haverá vários cenários de teste, e a entrada termina por EOF.
Cada caso de teste é formado por apenas uma linha:

Nesta linha haverá dois números inteiros A e B ($10 \leq A \leq B \leq 10^5$), que são o começo e o fim do intervalo a ser considerado respectivamente.

Obs.: A e B não devem ser considerados.

Saída

Para cada caso de teste seu programa deve imprimir o menor e maior número palíndromo no intervalo, ou -1 caso eles não existam.

Exemplo de entrada (Caso de teste 1)	Exemplo de saída (Caso de teste 1)
10 25 10 114 12 40 200 225 403 10560 50 54 10 100 1600 1800 2942 24450	11 22 11 111 22 33 202 222 404 10501 -1 11 99 1661 1771 2992 24442

Problema H
Lógica

Arquivo fonte: logica.c, logica.cpp ou logica.java

Você já ouviu falar na lógica proposicional? É um sistema formal no qual as fórmulas representam sentenças que são formadas pela combinação de proposições atômicas usando conectivos lógicos.

Uma proposição atômica é uma declaração afirmativa à qual se pode associar um valor verdadeiro (true) ou falso (false), mas não ambos. Por exemplo, "O Brasil fica na América" (A) é uma proposição verdadeira, enquanto "A lua é de queijo" (B) é uma proposição falsa.

Já os conectivos lógicos são símbolos que conectam duas proposições. Por exemplo, o conectivo "E" (&) e o conectivo "OU" (|). Assim, seja a sentença: "O Brasil fica na América **E** a lua é de queijo". Ela pode ser representada pela seguinte fórmula: A&B. Enquanto a sentença "O Brasil fica na América **OU** a lua é de queijo" corresponde a A|B.

Sua tarefa nesse problema é desvendar fórmulas lógicas, ou seja, dizer se uma dada fórmula (conjunto de proposições) é verdadeira (true) ou falsa (false).

Considere que os únicos conectivos lógicos existentes são "E" e "OU" e que possuem a seguinte semântica:

E (&)				OU (|)		
A	**B**	**Resultado**		**A**	**B**	**Resultado**
True	True	True		True	True	True
True	False	False		True	False	True
False	True	False		False	True	True
False	False	False		False	False	False

Considere também a existência de parêntesis nas fórmulas para indicar precedência, assim como numa expressão matemática. Quando não houver parêntesis, os conectivos "E" e "OU" possuem precedência igual, sendo a fórmula resolvida sempre da esquerda para a direita. Exemplos:

```
1) F&((T|F)|(F&F)) = F
2) F&F&F|T = T
3) ((F|F|T)&(T&F)|T) = T
```

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste é uma fórmula lógica de tamanho N (3 ≤ N ≤ 1000), contendo os símbolos 'F', 'T', '(', ')', '&', '|'. Não existe caracter de espaço separando os símbolos.
O final da entrada é indicado por final de arquivo (EOF).

Saída

Para cada caso de teste, imprima "T" (sem aspas) se o resultado da fórmula for verdadeiro (true) ou "F" (sem aspas) se o resultado da fórmula for falso (false).

Exemplos de Entrada

```
(((F&T))|T)
(((F|F|T)&(T&F))|T)
((F|F|T)&(T&F)|T)
F&F&F
```

Exemplos de Saída

```
T
T
T
F
```

MARATONA DE PROGRAMAÇÃO

Problema I

Ana

Arquivo fonte: ana.c, ana.cpp ou ana.java

A jovem e perspicaz Ana Karina é encantada com os palíndromos e sua simetria. Ela apenas acha estranho que as palavras que a originaram não sejam simétricas, já que em grego *palin* significa "trás" e *dromos* significa "corrida".

Apesar de seu interesse, Ana ainda não possui um programa que verifica se uma sequência é palíndromo, ou não.

Ajude Ana, sabendo que, normalmente, nos palíndromos são desconsiderados os sinais ortográficos (ex.: acentos e de pontuação) e os espaços entre as palavras. Como também não são relevadas letras maiúsculas e minúsculas.

Entrada

A entrada para o programa contém vários casos de teste, um por linha, onde cada entrada possui n caracteres, tal que 1 <= n <= 400.
A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada produzir uma linha na saída com "sim", caso a entrada seja palíndromo, e "nao", caso não seja.
A saída deve ser escrita na saída padrão.

Exemplo de entrada

Ovo!
Ande, Edna.
Ipameri!
Ema
Eu, ué!

Saída para o exemplo de entrada

sim
sim
nao
nao
sim

MARATONA DE PROGRAMAÇÃO

Problema J
Idade

Arquivo fonte: idade.c, idade.cpp ou idade.java

Um desenvolvimento com foco automático possibilita evitar alguns erros humanos, como por exemplo, no cálculo de datas ao preencher formulários. Neste preenchimento pode haver problemas como o do cálculo de idade, mas há formas de prevenir isto.

Entrada

A primeira linha de entrada fornece um número de casos T (1 <= T <= 200), onde T indica o número de casos de teste. Cada caso de teste inicia com uma linha branca, seguida de 2 linhas com a data corrente e a data de aniversário, respectivamente. As datas estão no formato: DD/MM/YYYY, onde DD é o Dia, MM o mês e YYYY o ano. Todas as datas serão válidas.
A entrada deve ser lida da entrada padrão.

Saída

A saída possui uma linha por caso de teste e terá uma (apenas uma) das 3 opções seguintes:

- "data de aniversário inválida", se a idade calculada for impossível (ainda não tiver nascido).
- "verifique a data de aniversario", se a idade calculada é maior que 130.
- A idade calculada (apenas em anos).

A saída deve ser escrita na saída padrão.

Exemplo de entrada

```
4

01/01/2007
10/02/2007

09/06/2007
28/02/1871

12/11/2007
01/01/1984

28/02/2005
29/02/2004
```

Saída para o exemplo de entrada

```
data de aniversário inválida
verifique a data de aniversário
23
0
```