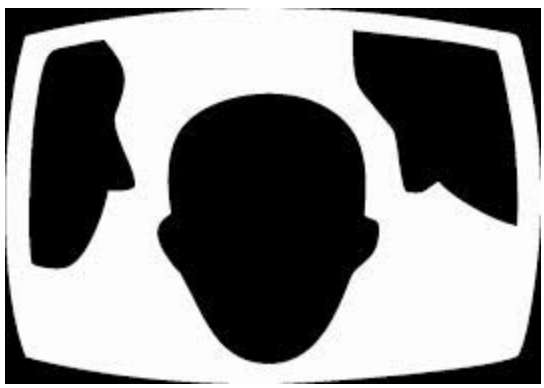
V Maratona de Programação

Leia as instruções com atenção

- O endereço para acessar o sistema de submissão é : http://boca.cct.uenp.edu.br
- Utilize o *login* e senha fornecidos para sua equipe.
- Caso precise trocar de computador, informar aos fiscais de sala para que providenciem um novo computador.
- Este caderno contém 10 exercícios.
- Cada exercício tem seu enunciado, formas de entrada e saída de dados e exemplos. Além disso, cada exercício tem um nome previamente escolhido. Lembre-se de salvar o arquivo com o nome indicado logo abaixo do título do exercício.
- Todos os exercícios seguem o padrão de uma saída por entrada: para cada entrada completa, o programa deve mostrar o resultado e finalizar.
- Considere que as entrada são obtidas por meio do teclado e que a saída é exibida no monitor.
- É estritamente proibido acesso a qualquer aparelho eletrônico que não seja o computador que a equipe está usando e *pen drive*.
- É permitido o acesso a qualquer material escrito ou impresso, inclusive livros e apostilas.
- Dúvidas poderão ser submetidas aos juizes por meio do botão “clarification” no próprio ambiente. No entanto, não serão esclarecidas dúvidas sobre o enunciado das questões ou sobre a solução, tais como o porquê de um determinado problema ter sido recusado.
- Se tiver algum problema durante a prova, chame um dos fiscais de sala. No entanto, os fiscais não esclarecerão dúvidas referentes ao enunciado ou à solução do exercício.
- Para cada submissão rejeitada, haverá uma penalidade de 10 minutos.



Bandeirantes, 18 de junho de 2016

Escriba

Arquivo: `Escriba.c`, `Escriba.cpp` ou `Escriba.java`

A Associação dos Programadores Apaixonados (APA) é uma sociedade sem fins lucrativos que realiza inúmeras reuniões para realizar debates filosóficos sobre os *bugs* em programas. Pelas regras da APA, o que é conversado nas reuniões deve ser transcrito para uma ata e ser assinado pelos presentes. No entanto, nenhum dos associados quer transcrever as conversas, porque acham isso tediante. Por isso resolveram pagar o desenvolvimento de um software que "ouça" o que é falado e gere um arquivo texto com tudo que é dito pelos associados. Você e dois amigos resolveram desenvolver o software e receber uns trocados para ~~a cerva~~ o pastel do fim de semana. No entanto, o software apresenta um pequeno problema. Ele insere os seguintes caracteres estranhos no meio do texto: **##**, **$$** e **&&**. Como já ocorreram várias reuniões e foram geradas várias atas, sua tarefa é desenvolver um programa que elimine estes caracteres e gere um texto correto, enquanto o defeito no software não for corrigido. (Observe que a presença de um dos caracteres isoladamente não é um erro, apenas considere incorreto quando aparecer mais de um juntos).

Entrada

A entrada contém uma única linha de texto com N caracteres (1 < N < 1000) contendo caracteres estranhos.

Saída

A saída contém uma linha de texto sem os caracteres incorretos.

Exemplos de entrada e saída

Exemplo 1

`Se as varia##veis tivere&&m uma q$$uantidade par de letras`	`Se as variaveis tiverem uma quantidade par de letras`

Exemplo 2

`Em P##HP, variaveis locais&& comecam com o caractere $.`	`Em PHP, variaveis locais comecam com o caractere $.`

Entradas Falsas

Arquivo: `Entradas.c`, `Entradas.cpp` ou `Entradas.java`

Seu time organizou uma festa para comemorar a vitória na V Maratona de Programação. Todos os alunos da UENP foram convidados para a noite, que incluía coquetel, jantar e uma sessão de exibição em telões dos códigos-fonte dos programas criados pelo seu time. O evento foi um sucesso: os convidados puderam debater sobre a alta qualidade dos algoritmos. No entanto, alguns críticos disseram que tamanho público esteve presente na festa graças à comida e não graças aos programas.

Independentemente da razão, no dia seguinte você descobriu o motivo pelo qual havia tantas pessoas: alguém lhe contou que diversos dos ingressos usados pelos presentes eram falsos. O número real de bilhetes foram numerados sequencialmente de 1 a N (N < 10000). Suspeita-se que algumas pessoas usaram o *scanner* e a impressora de uma república suspeita para produzir cópias dos bilhetes verdadeiros. Para determinar o tamanho do prejuízo, você deve utilizar amostras (um pacote de ingressos coletados na entrada da festa). Você deve determinar quantos bilhetes no pacote contém "clones": outro bilhete com o mesmo número da sequência.

Entrada

A entrada contém dados de diversos casos de teste. Cada caso de teste contém duas linhas. A primeira linha contém dois inteiros **N** e **M**, que indicam, respectivamente, o número de bilhetes originais e o número de pessoas presentes na festa (1 ≤ **N** ≤ 10000 e 1 ≤ **M** ≤ 20000). A segunda linha do caso de teste contém **M** inteiros $\mathbf{T}_i$ representando os números dos bilhetes contidos no pacote utilizado como amostra (1 ≤ $\mathbf{T}_i$ ≤ **N**). O final da entrada é indicado por **N** = 0 e **M** = 0.

Saída

Para cada caso de teste seu programa deverá imprimir uma linha, contendo a quantidade de bilhetes do pacote que contém outro bilhete com o mesmo número da sequência.

Exemplos de entrada e saída

Exemplo

5 5 3 3 1 2 4 6 10 6 1 3 6 6 4 2 3 1 2 0 0	 1 4

Texto em Triângulo

Arquivo: `Triangulo.c`, `Triangulo.cpp` ou `Triangulo.java`

Crie um programa que receba uma frase de tamanho indeterminado e escreva o texto em forma de um triângulo. As linhas do triângulo possuem apenas um espaço entre cada palavra e são alinhadas à esquerda. Cada linha do triângulo deve ser menor que a anterior para garantir o seu formato. A primeira linha é a maior enquanto a última linha possui apenas a última palavra da entrada. O texto será informado sem espaços duplicados.

Entrada

A entrada contém uma sequência de **n** caracteres (1 < n < 1000000) contendo caracteres alfanuméricos, símbolos e pontuações.

Saída

Para cada caso de teste seu programa deverá imprimir o texto em formato de triângulo de modo que a primeira linha seja a com maior quantidade de palavras, as linhas subsequentes devem conter uma quantidade menor de palavras que a anterior e a última linha contenha apenas uma palavra. Os sinais de pontuação são considerados na contagem de caracteres e as palavras não podem ser separadas.

Exemplos de entrada e saída

Entrada

```
The Free software is a matter of liberty not price. To understand the concept you should think of free as in free speech not as in free beer.
```

Saída

```
The Free software is a matter of liberty
not price. To understand the
concept you should think
of free as in free
speech not as
in free
beer.
```

Entrada

```
The development of GNU, started in January 1984, is known as the GNU Project. Many of the programs in GNU are released under the auspices of the GNU Project; those we call GNU packages.
```

Saída

```
The development of GNU started in January 1984
is known as the GNU Project.
Many of the programs in GNU
are released under the
auspices of the GNU
Project; those
we call GNU
packages.
```

Abominável Cifra das Neves

Arquivo: `Cifra.c`, `Cifra.cpp` ou `Cifra.java`

O estudante Eduardo Nevesden, preocupado com sua segurança, pensou em uma maneira de codificar as informações que envia para os amigos pela internet. Para isso ele pensou em uma variação da cifra de César baseada em uma sequência de chaves.

A codificação da cifra de César consiste em, dado um valor inteiro para a chave, por exemplo 3, o texto deve ser codificado contando as próximas 3 letras do alfabeto.

Exemplo:

Chave = 3
Texto = MARATONA
Texto Codificado= PDUDXRQD

Contudo, Eduardo pensou em uma criptografia um pouco mais complexa que funciona da seguinte maneira:

- O programa recebe duas entradas, sendo a primeira uma sequência de 9 números inteiros entre 0 e 9, e a segunda entrada o texto a ser codificado.
- Cada letra do texto deverá ser codificado com a cifra de César utilizando o número na posição correspondente da chave informada.
- Se o texto for maior que o tamanho da sequência de chaves, esta é aplicada novamente a partir do início.

Exemplo:

Chave = 456789123
Texto = PROGRAMACAO

Aplicação:

456789123 456789123
PROGRAMAC AO

Resultado:

TWUNZJNCFET

Exemplos de entrada e saída

Exemplo

456789123 PROGRAMACAO 123123111 MARATONA	TWUNZJNCFET NCUBVROB

Complexidade da Senha

Arquivo: `Senha.c`, `Senha.cpp` ou `Senha.java`

O estudante Eduardo Nevesden, além de utilizar a criptografia em suas comunicações na internet, também se preocupa em criar senhas complexas. Para fazer essa classificação, Eduardo classifica cada caractere da senha informada conforme a tabela abaixo:

Tipo	Valor	Padrão
Número	1	[0-9]
Letra minúscula	2	[a-z]
Letra maiúscula	3	[A-Z]
Caracteres Especiais	5	Qualquer outro

A complexidade é calculada pela soma dos valores de cada caractere da senha, denotado por x, e classificado conforme segue:

Classificação	Condição
FRACA	x>0 e x<=10
MEDIA	x>10 e x<=20
FORTE	x>20 e x<=50
SUPER_FORTE	x>50

Crie um programa que receba como entrada uma senha com tamanho máximo de 256 caracteres e calcule a sua complexidade, retornando o valor obtido e o texto com a classificação (FRACA, MEDIA, FORTE, SUPER_FORTE) separados por ponto e vírgula;

Exemplos de entrada e saída

Exemplo

123456 C0MPUT4C40 MAR4T0N4!!!UENP	6;FRACA 18;MEDIA 36;FORTE

Tautogramas

Arquivo: `Tautogramas.c`, `Tautogramas.cpp` ou `Tautogramas.java`

Fiona sempre amou poesia, e, recentemente, ela descobriu uma forma poética fascinante. Tautogramas são um caso especial de aliteração, que é a ocorrência da mesma letra, no início de palavras adjacentes. Em particular, uma sentença é um tautograma se todas as suas palavras começam com a mesma letra.
Por exemplo, as seguintes frases são tautogramas:

- Um único urso uivou
- usurpam usuários utópicos
- Trova Tróia, tudo truncado
- Sinto suave sensação

Fiona quer deslumbrar seu namorado com uma carta romântica cheia deste tipo de frases. Por favor, ajude Fiona para verificar se cada frase que ela escreveu a seguir é um tautograma ou não.

Entrada

Cada caso de teste é dado em uma única linha que contém uma frase. A sentença consiste de uma sequência de, no máximo, 50 palavras separadas por espaços simples. Uma palavra é uma sequência de no máximo 20 letras maiúsculas e minúsculas contíguas do alfabeto. Uma palavra contém pelo menos uma letra e uma frase contém, pelo menos, uma palavra.

Saída

Para cada caso de teste de saída de uma única linha contendo um maiúsculas 'S' se a sentença é um tautograma, ou uma letra maiúscula ‘N’ caso contrário.

Exemplos de entrada e saída

usurpam usuários utópicos	S
Trova Tróia, tudo truncado	S
É preciso saber viver	N
Sinto suave sensação	S
Receptividade regride raiva	S
A dor vai curar essas lástimas	N
temo ter temperamento	S
totalmente transitório	S

Digitos Diferentes

Arquivo: `Digitos.c`, `Digitos.cpp` ou `Digitos.java`

Os habitantes de Nlogônia são muito supersticiosos. Uma das suas crenças é que os números das casas da rua que têm um dígito repetido trazer má sorte para os moradores. Portanto, eles nunca iriam viver em uma casa que tem um número rua como 838 ou 1004.

A rainha de Nlogônia ordenou a construção de uma nova avenida à beira-mar e quer atribuir às novas casas apenas números sem dígitos repetidos, para evitar desconforto entre os seus súditos. Você foi nomeado por Sua Majestade para escrever um programa que, dados dois inteiros **N** e **M**, determina o número máximo de casas que podem ser atribuídos números de rua entre **N** e **M**, inclusive, que não têm dígitos repetidos.

Entrada

Cada caso de teste é descrito usando uma linha. A linha contém dois inteiros N e M, como descrito acima (1 ≤ **N** ≤ **M** ≤ 5000).

Saída

Para cada caso de teste de saída uma linha com um inteiro representando a quantidade de números de casa da rua entre **N** e **M**, inclusive, sem dígitos repetidos.

Exemplos de entrada e saída

Exemplo

87 104 989 1022 22 25 1234 1234	14 0 3 1

Histórico de Comandos

Arquivo: `Historico.c`, `Historico.cpp` ou `Historico.java`

Uma interface por linha de comando (ILC) é um dos tipos de interface humano-computador mais antigos que existem. Uma ILC permite a interação com o software através de um interpretador de comandos, sendo normalmente acessível em um terminal (ou janela) de texto. A vantagem de um interpretador de comandos é que ele permite que o usuário opere o sistema usando apenas o teclado. Ainda hoje em dia, em que estamos acostumados com interfaces gráficas sofisticadas, muitos aplicativos e sistemas operacionais incluem algum tipo de interface por linha de comando, e muitos usuários ainda preferem usá-la para grande parte das tarefas. Um dos recursos mais úteis de um interpretador de comandos é o histórico de comandos.

Quando um comando é digitado e executado, ele é colocado no histórico de comandos do terminal. O comando pode ser exibido novamente no terminal apertando a tecla '↑'; a tecla Enter executa o comando novamente quando o comando está sendo exibido no terminal. Todos os comandos executados são guardados no histórico: pressionar a tecla '↑' duas vezes exibe o penúltimo comando executado, pressioná-la três vezes exibe o antepenúltimo comando, e assim sucessivamente.

Por exemplo, se o histórico inicial é (A, B, C, D), para repetir o comando C basta pressionar duas vezes a tecla '↑'. O histórico será então atualizado para (A, B, C, D, C). Nesse ponto, para repetir o comando A será necessário pressionar cinco vezes a tecla '↑'; o histórico será atualizado para (A, B, C, D, C, A). Nesse ponto, para repetir mais uma vez o comando A basta pressionar uma vez a tecla '↑'; o histórico será atualizado para (A, B, C, D, C, A, A).

Leandro é administrador de sistemas e usa frequentemente o interpretador de comandos para gerenciar remotamente os servidores que administra. Em geral, ele precisa apenas repetir comandos que já havia digitado antes. Enquanto estava trabalhando em um servidor, ele teve uma curiosidade: quantas vezes ele precisa pressionar a tecla '↑' para executar uma determinada sequência de comandos? Ele sabe quais são as posições no histórico dos comandos que ele necessita executar, mas não sabe resolver esse problema. Por isso, pediu que você fizesse um programa que respondesse à pergunta dele..

Entrada

A entrada é composta de vários casos de teste. A primeira linha de cada caso de teste contém um número inteiro N , indicando o número de comandos que Leandro deseja executar ($1 \leq N \leq 1.000$). A segunda linha de um caso de teste contém N inteiros $P_1, P_2, ..., P_N$, que indicam as posições dos comandos no histórico ($1 \leq Pi \leq 1.000.000$) no momento inicial, na ordem em que os comandos devem ser executados. Ou seja, o primeiro comando que deve ser executado está inicialmente na posição P_1 do histórico; depois deve ser executado o comando que está inicialmente na posição P_2 no histórico, e assim por diante, até P_N , que é a posição inicial do último comando que deve ser executado. Note que pode haver $P_i = P_j$. As posições são dadas em função do número de vezes que a tecla '↑' deve ser pressionada: um comando na posição 5 necessita que a tecla '↑' seja pressionada cinco

vezes antes de aparecer no terminal (note que à medida que comandos vão sendo executados, a posição de um dado comando no histórico pode mudar)..

Saída

Para cada caso de teste, seu programa deve imprimir apenas uma linha, contendo o número de vezes que Leandro precisa pressionar a tecla '↑' para executar todos os comandos.

Exemplos de entrada e saída

3 2 5 3 4 2 1 4 3 5 1 2 3 4 5 4 1 3 1 3	13 16 25 9

Pôneis Malditos

Arquivo: `Poneis.c`, `Poneis.cpp` ou `Poneis.java`

A Associação dos Criadores de Pôneis (ACP) é uma sociedade que hospeda estes animais em seu hotel de pôneis. Acontece que, por uma razão ainda não sabida, alguns associados gostam de criar uma estranha raça de pôneis que é maldita. Uma característica de um pônei maldito é que ele não pode ser hospedado ao lado de um pônei comum. Se isto acontecer, o pônei comum enlouquece. No entanto, é interessante para a ACP hospedar pôneis malditos porque eles pagam o dobro do preço. No momento, eles estão com dificuldade de gerenciar as baias do hotel, para impedir que um pônei maldito fique ao lado de um pônei comum. Eles ficaram sabendo que seu time participou da Maratona de Programação da UENP e pediram ajuda para o desenvolvimento de um programa para gerenciar as baias, como segue.

Inicialmente o programa deve ler um inteiro X que representa a quantidade de baias do hotel. A seguir, são lidos N caracteres que podem ser 'C', indicando que um pônei comum quer se hospedar ou 'M' indicando que é um pônei maldito. Para cada entrada, o programa deve dizer "sim" (caso seja possível hospedar) ou "não" (caso não seja possível). Deve ser possível hospedar um pônei comum quando houver baia livre e o pônei não fique ao lado de um pônei maldito. Deve ser possível hospedar um pônei maldito quando houver baia livre que não esteja ao lado de um pônei comum. Um exemplo de configuração em dado momento para as baias seria:

C		M	M	M		C	C		M

em que há 10 baias nas quais estão hospedados 3 pôneis comuns e 4 pôneis malditos.

Entrada

A entrada contém um inteiro X (0 < X < 1000) contendo a quantidade de baias, seguidos de caracteres 'C' ou 'M'. Encerre a digitação quando for digitado o caractere '#'.

Saída

A saída contém a mensagem "sim" ou "nao", indicando se o pônei pode ou não ser hospedado.

Exemplos de entrada e saída

Exemplo 1

10 C M M M C C M C M #	 sim sim sim sim sim sim sim nao nao

Exemplo 2

3 C C M C C #	 sim sim nao sim nao

Ingressos

Arquivo: `Ingressos.c`, `Ingressos.cpp` ou `Ingressos.java`

Sua turma está organizando uma festa para arrecadar dinheiro para formatura. No entanto, um dos membros da equipe organizadora é muito supersticioso: não gosta do número "4" nem da sequência de dígitos "13". Por isso, ele quer que os ingressos para festa sejam numerados de tal forma que não tenham o número 4 nem nem o 13. Por exemplo, não deve haver o ingresso 4, nem o 13, nem o 134, mas pode haver o 103. Assim, os primeiros ingressos numerados seriam: 1, 2, 3, 5, 6 ,7, 8, 9, 10, 11, 12, 15, 16,...

Como os ingressos da festa serão muitos, sua equipe deve elaborar um programa que, dado a quantidade de ingressos a ser emitida, informe o número do último ingresso.

Entrada

Cada caso de teste consiste em uma única linha, contendo um inteiro N (1 <= N <= 100000) que indica a quantidade de ingressos a ser emitida.

Saída

Para cada caso de teste, imprima uma linha contendo um único inteiro indicando o número do último ingresso emitido.

Exemplos de entrada e saída

Exemplo 1

15	18

Exemplo 2

25	29