Lista-03

Prof. Msc. Elias Batista Ferreira
Prof. Dr. Gustavo Teodoro Laureano
Profa. Dra. Luciana Berretta
Prof. Dr. Thierson Rosa Couto

Sumário

1 Achei (+)

 (+)

Faça um programa que receba um vetor V com N números inteiros e posteriormente receba M números e verifique se eles estão ou não no vetor.

Entrada

O programa terá apenas um caso de teste. Na primeira linha do caso de teste há um número inteiro N, $1 \leq N \leq 100000$, representando o tamanho do vetor V. Na linha seguinte haverá N números inteiros separados por um espaço em branco, que são nos N valores do vetor V. Na terceira linha será informado um número inteiro M, $1 \leq M \leq 1000$, representando a quantidade de buscas que serão efetuadas no vetor. Logo em seguida haverá M linhas, cada uma com um número inteiro que deve ser buscado no vetor V.

Saída

Seu programa gera M linhas de saída. Cada uma com o resultado da Busca dos M números inteiros no vetor V. Quando o valor estiver no vetor V escreva "ACHEI", quando não estiver escreva "NAO ACHEI", com todas as letras maiúsculas e sem acentos. Ao final quebre uma linha.

Exemplo

Entrada
10 9 0 1 3 8 2 7 4 6 5 4 1 23 4 7
Saída
ACHEI NAO ACHEI ACHEI ACHEI

2 Contagem (+)

 (+)

Dado um vetor V de tamanho N e um inteiro K, contabilize quantos elementos de V são maiores ou iguais ao inteiro K.

Entrada

O programa terá apenas um caso de teste. O programa deve ler, obrigatoriamente, um número N que pertença ao intervalo $1 \leq N \leq 1000$. Se N lido não for válido, o programa deve fazer uma nova leitura de N. Caso N seja válido, N representa o tamanho do vetor V. Na próxima linha há N números inteiros separados por um espaço em branco cada, representando cada elemento do vetor V. E finalmente, na última linha há um inteiro K.

Saída

Seu programa gera apenas uma linha de saída contendo um número inteiro representando quantos elementos do vetor V são maiores ou iguais ao inteiro K. Após a impressão do valor quebre uma linha.

Exemplo

Entrada
0 -3 4 1 2 3 4 0
Saída
4

Entrada
10 1 2 3 4 5 6 7 8 9 10 5
Saída
6

Entrada
10 1 2 3 4 5 6 7 8 9 10 20
Saída
0

Entrada
1 2 3
Saída
0

Entrada
4 1 4 6 4 4
Saída
3

3 Imprimir Um Vetor na Ordem Inversa (+)

 (+)

Escreva um programa em C para armazenar n valores inteiros em um vetor, e depois imprimi-los na ordem inversa a qual foram lidos.

Entrada

A entrada contém duas linhas. A primeira, contém um valor inteiro $n < 5000$ que corresponde ao número de elementos que aparecem na segunda linha. A segunda linha contém n valores inteiros, separados entre si por um espaço.

Saída

A saída é formada por uma linha contendo os n na ordem inversa da qual foram lidos.

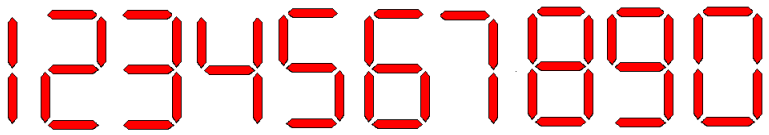
Exemplo

Entrada
7 3 6 2 9 2 7 9
Saída
9 7 2 9 2 6 3

4 LED

 (+)

João quer montar um painel de leds contendo diversos números. Ele não possui muitos leds, e não tem certeza se conseguirá montar o número desejado. Considerando a configuração dos leds dos números abaixo, faça um algoritmo que ajude João a descobrir a quantidade de leds necessário para montar o valor.



Entrada

A entrada contém um inteiro $N, (1 \leq N \leq 1.000)$ correspondente ao número de casos de teste, seguido de N linhas, cada linha contendo um número $(1 \leq V \leq 10^{100})$ correspondente ao valor que João quer montar com os leds.

Saída

Para cada caso de teste, imprima uma linha contendo o número de leds que João precisa para montar o valor desejado, seguido da palavra "leds".

Exemplo

Entrada
3 115380 2819311 23456
Saída
27 leds 29 leds 25 leds

5 Quantas Letras?

 (+)

Tia Magnólia está ensinando as crianças a reconhecerem letras, e entre as letras quais são vogais e quais são consoantes. Ela precisa fazer vários testes com seus alunos. Ela quer que eles leiam várias linhas de um texto e contem em cada linha quantas letras (maiúsculas ou minúsculas), quantas vogais (maiúsculas ou minúsculas) e quantas consoantes (minúsculas ou maiúsculas) existem em cada linha lida. Como Tia Magnólia possui vários textos, ela gostaria de uma forma automatizada de obter essa contagem para gerar um gabarito que permita a ela verificar se as respostas dos alunos estão corretas ou não. Sabendo que você é "FERA" em processamento de strings, ela quer que você faça um programa que gere essas contagens para ela.

Entrada

A entrada contém vários casos de teste. A primeira linha contém um inteiro N que indica a quantidade de casos de teste. Cada caso de teste consiste de uma única linha de texto. A linha pode conter qualquer tipo de caractere (letras e "não letras"). Uma linha pode conter até 10.000 caracteres.

Saída

Para cada caso de teste, imprima três mensagens, cada uma em uma linha diferente. A primeira mensagem deve estar no seguinte formato: "Letras = x". A segunda mensagem deve ser : "Vogais = y" e a última mensagem deve ser: "Consoantes = z". Os valores de x, y e z nas mensagens correspondem aos totais de, respectivamente, letras, vogais e consoantes encontrados em um caso de teste.

Exemplo

Entrada
4 Este e um caso de teste dos varios possiveis Vem ver vovo! O presidente renunciou? #chapeuzinho#Vermelho#
Saída
Letras = 36 Vogais = 17 Consoantes = 19 Letras = 10 Vogais = 4 Consoantes = 6 Letras = 20 Vogais = 10 Consoantes = 10 Letras = 19 Vogais = 8 Consoantes = 11

6 Um_Dois_Três

 (+)

Seu irmão mais novo aprendeu a escrever apenas um, dois e três, em Inglês. Ele escreveu muitas dessas palavras em um papel e a sua tarefa é reconhecê-las. Nota-se que o seu irmão mais novo é apenas uma criança, então ele pode fazer pequenos erros: para cada palavra, pode haver, no máximo, uma letra errada. O comprimento de palavra é sempre correto. É garantido que cada palavra que ele escreveu é em letras minúsculas, e cada palavra que ele escreveu tem uma interpretação única.

Entrada

A primeira linha contém o número de palavras que o seu irmão mais novo escreveu. Cada uma das linhas seguintes contém uma única palavra com todas as letras em minúsculo. As palavras satisfazem as restrições acima: no máximo uma letra poderia estar errada, mas o comprimento da palavra está sempre correto. Haverá, no máximo, 1000 palavras de entrada.

Saída

Para cada caso de teste, imprima o valor numérico da palavra

Exemplo

Entrada
3 owe too theee
Saída
1 2 3

7 Acumulado de Elementos (++)

 (++)

Faça um programa que receba vários vetores de números inteiros, calcule o maior elemento (M) de cada vetor e apresente a frequência dos valores menores ou iguais a i, onde $i = 0, 1, 2, \ldots, M$.

Entrada

O programa possui vários casos de testes. A primeira linha de cada caso contém um inteiro $1 < N \leq 10000$, representando o tamanho do vetor. A segunda linha conterá N inteiros entre 0 e 10000, representando os N elementos do vetor. A entrada termina quando $N = 0$.

Saída

O programa gera várias linhas de saída para cada entrada. Cada linha apresenta o valor entre parênteses seguido de um espaço em branco e a quantidade de números que são menores ou iguais a esse valor, seguido de '\n'

Exemplo

Entrada
10 6 13 7 3 13 6 14 3 14 9 5 9 8 7 6 5 8 0 1 2 3 4 5 6 7 0
Saída
(0) 0 (1) 0 (2) 0 (3) 2 (4) 2 (5) 2 (6) 4 (7) 5 (8) 5 (9) 6 (10) 6 (11) 6 (12) 6 (13) 8 (14) 10 (0) 0 (1) 0 (2) 0 (3) 0 (4) 0 (5) 1 (6) 2 (7) 3 (8) 4 (9) 5 (0) 1 (1) 2 (2) 3 (3) 4 (4) 5 (5) 6 (6) 7 (7) 8

8 Criptografia

 (++)

Solicitaram para que você construisse um programa simples de criptografia. Este programa deve possibilitar enviar mensagens codificadas sem que alguém consiga lê-las. O processo é muito simples. São feitas três passadas em todo o texto.

Na primeira passada, somente caracteres que sejam letras minúsculas e maiúsculas devem ser deslocadas 3 posições para a direita, segundo a tabela ASCII: letra 'a' deve virar letra 'd', letra 'y' deve virar caractere '|' e assim sucessivamente. Na segunda passada, a linha deverá ser invertida. Na terceira e última passada, todo e qualquer caractere a partir da metade em diante (truncada) devem ser deslocados uma posição para a esquerda na tabela ASCII. Neste caso, 'b' vira 'a' e 'a' vira '`'.

Por exemplo, se a entrada for "Texto #3", o primeiro processamento sobre esta entrada deverá produzir "Wh{wr #3". O resultado do segundo processamento inverte os caracteres e produz "3# rw{hW". Por último, com o deslocamento dos caracteres da metade em diante, o resultado final deve ser "3# rvzgV".

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém um inteiro $N(1 \leq N \leq 10^4)$, indicando a quantidade de linhas que o problema deve tratar. As N linhas contém cada uma delas $M(1 \leq M \leq 10^3)$ caracteres.

Saída

Para cada entrada, deve-se apresentar a mensagem criptografada.

Exemplo

Entrada

`4`
`Texto #3`
`abcABC1`
`vxpdylY .ph`
`vv.xwfxo.fd`

Saída

`3# rvzgV`
`1FECedc`
`ks.  \n{frzx`
`gi.r{hyz-xx`

9 Frequência e Maior (++)

 (++)

Dada uma sequência de N notas entre 0 e 10, escreva um programa que exiba o valor da última nota informada e quantas vezes ela apareceu no conjunto. O programa deve exibir ainda a maior nota informada e a posição (índice do vetor) da sua primeira ocorrência.

Entrada

Na primeira linha há um inteiro N, sendo $1 \leq N \leq 10000$ representando a quantidade de notas da sequência. Não é necessário validar o valor de N na entrada. Nas N linhas seguintes haverá um número inteiro entre 0 e 10, inclusive, em cada linha.

Saída

O programa gera 2 linhas de saída. A primeira linha exibirá a frequência da última nota informada e a segunda linha exibirá a maior nota e a posição (índice do vetor) da sua primeira ocorrência, seguindo o formato da saída apresentado a seguir. Não se esqueça de quebrar uma linha após a última impressão.

Exemplo

Entrada
11 5 6 3 4 3 8 7 4 8 6 4
Saída
Nota 4, 3 vezes Nota 8, indice 5

10 Inverte Vetor (++)

 (++)

Faça um programa que receba um vetor V de N inteiros e construa um vetor W com os mesmos elementos de V, porém invertidos, mostre os vetores V e W e o maior elemento de V e o menor elemento de W.

Entrada

A entrada contém apenas um caso de teste com 2 linhas. Na primeira linha há um inteiro N, $1 < N \leq 1000$, representando o tamanho do vetor V. Na segunda linha há N valores inteiros separados por um espaço em branco cada, que são os valores do vetor V.

Saída

O programa deve gerar 4 linhas de saída. A primeira linha deve haver N inteiros separados por um espaço em branco cada, representando os elementos do vetor V. Atenção, após o último elemento de V não deve haver um espaço em branco. A segunda linha deve haver N inteiros separados por um espaço em branco cada, representando os elementos do vetor W. Atenção, após o último elemento de V não deve haver um espaço em branco. A terceira linha deve haver apenas um inteiro, representando o maior elemento de V. A quarta linha deve haver apenas um inteiro, representando o menor elemento de W. Após imprimir a quarta linha da saída, quebre uma linha.

Exemplo

Entrada
5 7 8 4 9 2
Saída
7 8 4 9 2 2 9 4 8 7 9 2

Entrada
8 235 6 23 5 78 123 89 4
Saída
235 6 23 5 78 123 89 4 4 89 123 78 5 23 6 235 235 4

Entrada
10 1 2 3 4 5 6 7 8 9 0
Saída
1 2 3 4 5 6 7 8 9 0 0 9 8 7 6 5 4 3 2 1 9 0

11 Mediana (++)

 (++)

Em teoria da probabilidade e estatística, a mediana, é uma medida de localização do centro da distribuição dos dados, definida do seguinte modo: Ordenados os elementos da amostra, a mediana é o valor (pertencente ou não à amostra) que a divide ao meio, isto é, 50% dos elementos da amostra são menores ou iguais à mediana e os outros 50% são maiores ou iguais à mediana. Para uma coleção de tamanho ímpar, a mediana é exatamente o elemento médio, ou seja, aquele que a divide de acordo com a regra citada. Já para uma coleção de tamanho par, a mediana é determinada como a média aritmética dos dois elementos centrais.

Entrada

A entrada consiste de um único caso de teste. Na primeira linha, é informado um inteiro N, $0 < N \leq 10^6$, representando a quantidade de elementos da amostra de dados. Nas N linhas seguintes é informado um inteiro por linha, este valor varia de -2^{32} a $2^{32} - 1$.

Saída

A saída consiste da mediana dos dados informados. O valor da mediana deve ser formatado com duas casas decimais.

Exemplo

Entrada
6 1 3 4 5 4 2
Saída
3.50

Entrada
7 3 9 1 5 4 7 1
Saída
4.00

12 Prefixo de Uma String

Escreva um programa para ler várias linhas na entrada. Cada linha contém um número inteiro seguido por um espaço e por uma string. Para cada linha, o programa deve chamar uma função do tipo **ponteiro para char** que que receba como primeiro parâmetro *n* o inteiro lido e como segundo parâmetro *s* a string lida. A função deve alocar espaço suficiente para armazenar os *n* primeiros caracteres de *s* (prefixo de *s*). Deve copiar os *n* primeiros caracteres de *s* para essa nova string e retornar o endereço da string criada. Se *n* for maior que o tamanho da string *s*, o prefixo corresponde a uma cópia da string *s*. A função deve retornar NULL, se não conseguir alocar o espaço necessário para um prefixo. Após chamar a função, o programa deve verificar se função retornou um endereço válido de prefixo, e nesse caso, deve imprimir o prefixo e deve liberar a área ocupada pelo prefixo, antes de processar uma nova linha

Entrada

A primeira linha da entrada contém um inteiro positivo $N(1 \leq N \leq 20)$, o qual corresponde ao número de casos de teste. Cada caso de teste corresponde a uma linha. Cada linha possui um número inteiro positivo *n*, um espaço e uma string *s*, com no máximo 499 caracteres.

Saída

Para cada caso de teste o programa deve imprimir uma linha contendo o prefixo de tamanho *n* da string *s* lida naquele caso de teste.

Exemplo

Entrada:
5 1 Universidade Federal de Goias 0 Introducao a Programacao 3 Universidade Federal de Goias 20 Universidade Federal de Goias 30 Universidade Federal de Goias

Saída:
U Uni Universidade Federal Universidade Federal de Goias

13 Sequência Espelho

 (++)

Imprimir números em sequência é uma tarefa relativamente simples. Mas, e quando se trata de uma sequência espelho? Trata-se de uma sequência que possui um número de início e um número de fim, e todos os números entre estes, inclusive estes, são dispostos em uma sequência crescente, sem espaços e, em seguida, esta sequência é projetada de forma invertida, como um reflexo no espelho. Por exemplo, se a sequência for de 7 a 12, o resultado ficaria 789101112211101987.

Entrada

A entrada possui um valor inteiro C indicando a quantidade de casos de teste. Em seguida, cada caso apresenta dois valores inteiros, B e E $(1 \leq B \leq E \leq 12221)$, indicando o início e o fim da sequência.

Saída

Para cada caso de teste, imprima a sequência espelho correspondente.

Sugestão

Utiliza a função sprintf() para imprimir um número inteiro em uma string. Use a função strlen() para obter o tamanho de uma string.

Exemplo

Entrada
3 1 5 10 13 98 101
Saída
1234554321 1011121331211101 98991001011010019989

14 Lê *strings* (+++)

 (+++)

Essa questão é opcional.

"Chega!!! Estou farto dos 'problemas' do `scanf` com o '%s'. Preciso de uma função mais estável para usar em meus códigos" . Já é sabido que a função `scanf` para ler *strings* apresenta problemas por conta de caracteres residuais de outras leituras no terminal. Ao pressionarmos ENTER no terminal, são enviados dois caracteres, o "\r\n", com os respectivos códigos ASCII: 10, 13, onde '\n' indica o final de texto enviado pelo terminal. Por exemplo, ao digitarmos o texto "123"+ENTER no terminal, a sequência de caracteres passada é "123\r\n".

A função `scanf` processa o texto enviado pelo terminal de acordo com o código do formato passado como parâmetro. O '\n' é consumido e o texto residual "123\r" é processado pelo `scanf`. Ao processar o texto "123\r"com o código de formado "%d", a função `scanf` busca de uma sequência de caracteres da esquerda para a direita que representa um número inteiro até encontrar um caracter que não seja um dígito. O processamento resulta no número 123 e sobra o caracter '\r', que irá compor o próximo texto a ser lido no terminal.

Ao digitar um novo texto, por exemplo, "abc"+ENTER, o texto enviado pelo terminal torna-se "\rabc\r\n". Aí começam os problemas... Ao ler um número inteiro ou real, o '\r' é consumido e ignorado, mas ao ler um caracter ou uma *string* o '\r' é considerado. Além disso, ele indica fim de texto para a leitura de *strings*. Para a leitura de caracteres e *strings*, as expressões "%c", "%s"e "%[^\n]"precisam lidar com o problema do '\r'. O código abaixo, por exemplo, ao receber o texto "88"+ENTER via terminal, termina com a impressão "n: 88, c: 10", sem esperar pela leitura do caracter `ch`.

```
int n;
char ch;
scanf("%d", &n);
scanf("%c", &ch);
printf("n: %d, c: %d\n", n, ch);
```

Note que uma ligeira alteração no texto de entrada resolve esse problema para `ch`. Se informarmos o texto "88a"+ENTER via terminal, o código termina com a impressão "n: 88, c: 97", isso porque passamos o caracter 'a' antes do '\r'. O '\r' será um problema para a próxima leitura de um caracter ou *string*.

Uma alternativa a esse problema é incluir uma leitura de caracter adicional sempre que for necessário ler um caracter ou uma *string*, mas essa não é uma alternativa elegante.

Outra limitação do `scanf` é a característica de não ler espaços no início de uma *string*, o que pode ser um problema dependendo da aplicação que se deseja implementar. Além disso, não é seguro ler uma *string* uma vez que a função não sabe a quantidade de caracteres que o vetor pode armazenar.

Na verdade, as características do `scanf` citadas neste texto como "problemas"são comportamentos propositalmente implementados, que podem ser configurados para resolver esses e outros problemas mais complexos na leitura de *strings*.

Em vez de aprender a usar as expressões entendidas pelo `scanf`, faça uma função que leia *strings* de modo a não sofrer com problemas de caracteres residuais e que também permita a leitura de espaços. Adicionalmente, esse função deve retornar a quantidade de caracteres lidos e limitar a quantidade de carateres a serem lidos. Os caracteres "\r\n"não podem compor a *string* final. Lembre-se que uma *string* precisa do '\0' para indicar seu final. O protótipo da função deve ser o seguinte:

```
/**
 * @param str vetor de caracteres onde a string lida será gravada
 * @param n quantidade máxima de caracteres a ser lidos
 * @return quandidade de caracteres lidos
 */
int le_string( char * str, int n );
```

Seu programa principal deve ser o listado abaixo. A função `print_codes` imprime os códigos ASCII de cada caracter da *string* passada via parâmetro. Seu protótipo é:

```
/**
 * @param str string de entrada
 */
void print_codes( char * str );
```

Seu programa principal deve ser o listado abaixo. Esse código avalia se a função `le_string` consegue lidar com o caractere '\r'.

```
#define N 128+1

int main() {
   char str[N], s[N];
   char c;
   int i;

   scanf("%c", &c);
   le_string(str, 3);
   print_codes(str);
   printf("caracter:%c, str:%s\n", c, str);

   scanf("%c", &c);
   le_string(str, 5);
   print_codes(str);
   printf("caracter:%c, str:%s\n", c, str);

   scanf("%c", &c);
   le_string(str, 5);
   print_codes(str);
   printf("caracter:%c, str:%s\n", c, str);

   scanf("%d", &i);
   le_string(str, 3);
   print_codes(str);
   printf("inteiro:%d, str:%s\n", i, str);

   //printf("Digite inteiros separados por espaco: ");
   scanf("%d", &i);
   //printf("inteiro:%d\n", i);
   //printf("Le string (15):\n");
   le_string(str, 15);
   print_codes(str);
   printf("inteiro:%d, str:%s\n", i, str);

   //printf("Digite uma string sem espacos: ");
   scanf("%s", s);
   //printf("string:%s\n", str);
   //printf("Le string (10):\n");
   le_string(str, 100);
   print_codes(str);
   printf("string:%s, str:%s\n", s, str);

   //printf("Digite uma string com espacos: ");
   scanf("%s", s);
   //printf("string:%s\n", str);
   //printf("Le string (20):\n");
   le_string(str, 100);
   print_codes(str);
   printf("string:%s, str:%s\n", s, str);
```

```
51
52      return 0;
53  }
```

Entrada

- Um caracter + ENTER
- Texto com espaços + ENTER
- Sequência de caracteres sem espaços + ENTER
- Sequência de caracteres separados por espaços + ENTER
- Um número inteiro + ENTER
- Texto com espaços + ENTER
- Inteiros separados por espaços + ENTER
- Texto sem espaços + ENTER
- Texto com espaços + ENTER
- Texto com espaços + ENTER

Saída

Uma linha com os códigos de cada caractere da `str` lida pela função `le_string` seguida por uma linha com a apresentação do conteúdo do dado lido e da *string* `str`.

Observações

DICA: Você pode fazer uma função que leia caracter por caracter usando o "%c". Lembre-se que o texto passado pelo terminal é uma sequência de caracteres.

Exemplo

Entrada	Saída
x texto 123 abcdef x y z h w 99 texto com espacos 11 22 33 44 texto_sem_espacos texto2 com espacos Text with spaces	116,101,120 caracter:x, str:tex 98,99,100,101,102 caracter:a, str:bcdef 32,121,32,122,32 caracter:x, str: y z 116,101,120 inteiro:99, str:tex 32,50,50,32,51,51,32,52,52 inteiro:11, str: 22 33 44 116,101,120,116,111,50,32,99,111,109,32,101,115,112,97,99 string:texto_sem_espacos, str:texto2 com espacos 32,119,105,116,104,32,115,112,97,99,101,115 string:Text, str: with spaces

15 Aula Cancelada (+++)

 (+++)

Um professor X tem uma turma de N alunos. Frustrado com a falta de disciplina, ele decide cancelar a aula se menos de K alunos estão presentes quando a aula começa. Dado o tempo de chegada de cada aluno, determinar se a aula é cancelada. Caso a aula não seja cancelada, imprima uma lista com os alunos que chegaram antes do início da aula em ordem contrária à mostrada na entrada.

Entrada

A primeira linha apresenta dois números inteiros separados por um espaço: N (alunos da turma) e K (mínimo de presenças para que a aula não seja cancelada), com $0 \leq$ N, K, ≤ 1000. Na segunda linha há N inteiros separados por espaços (A1 , A2 ,... , An) descrevendo os tempos de chegada para cada aluno. Suponha que esta ordem seja a mesma da lista de presença do professor, com o primeiro aluno descrito na entrada sendo o aluno 1 e assim por diante. Nota: horários de chegada não-positivos (Ai ≤ 0) indicam que o aluno chegou cedo ou na hora; horários de chegada positivos (Ai>0) indicam o aluno chegou Ai minutos tarde.

Saída

O programa apresenta uma mensagem com a palavra "SIM" se a aula é cancelada, e "NAO" caso contrário. Após imprimir a mensagem quebre uma linha. Se a aula não for cancelada, imprima os M alunos presentes antes do início da aula (ou seja, com Ai ≤ 0) na ordem contrária da lista de entrada.

Exemplo

Entrada
4 3 -1 -3 4 2
Saída
SIM

Entrada
4 2 0 -1 2 1
Saída
NAO 2 1

Entrada
10 10 0 0 0 0 0 0 0 0 0 1
Saída
SIM

Entrada
2 1 -8 -4
Saída
NAO 2 1

Entrada
2 1 1 2
Saída
SIM

Entrada
10 4 -93 -86 49 -62 -90 -63 40 72 11 67
Saída
NAO 6 5 4 2 1

Entrada
10 10 23 -35 -2 58 -67 -56 -42 -73 -19 37
Saída
SIM

Entrada
10 1 88 -17 -96 43 83 99 25 90 -39 86
Saída
NAO 9 3 2

16 Contagem de Elementos Únicos (+++)

 (+++)

Elabore um programa que conte o número total de elementos únicos em um vetor de números inteiros.

Entrada

A entrada contém duas linhas. A primeira, contém um valor inteiro $n < 5000$ que corresponde ao número de elementos que aparecem na segunda linha. A segunda linha contém n valores inteiros, separados entre si por um espaço.

Saída

A saída é formada por uma linha contendo um valor inteiro que corresponde ao número de elementos que aparecem apenas uma vez no vetor. Após o valor, o programa deve imprimir o caractere de quebra de linha.

Exemplo

Entrada
7 3 6 2 9 2 7 9
Saída
3

17 Máxima Coordenada (+++)

 (+++)

Faça um programa que leia vários pares de pontos, calcule o vetor definido entre eles e imprima a coordenada do vetor que possui o maior valor absoluto (módulo). Considere que o vetor que liga dois pontos A (x1,y1,z1) e B (x2,y2,z2) é calculada como: BA = (x2 − x1, y2 − y1, z2 − z1)

Entrada

A entrada consiste de várias linhas. A primeira linha apresenta um número de pontos N, com $2 \leq N \leq 1000$. As N linhas seguintes apresentam pontos no espaço na forma x y z, com x, y e z números reais tais que $-1000 \leq x, y, z \leq 1000$. Faça um programa que calcule o vetor que liga dois pontos consecutivos nesta lista e imprima a coordenada de maior valor absoluto. Note que, com exceção do primeiro e último valor de entrada, todos os pontos serão utilizados duas vezes, uma para o cálculo do vetor com o ponto que veio antes na lista e outra para o ponto que veio depois.

Saída

A saída consiste de (N-1) linhas, cada uma contendo o módulo do valor da coordenada de maior valor absoluto, com 2 casas decimais após a vírgula. Após a impressão do último valor, quebre uma linha.

Exemplo

Entrada
2 4 1 0 -1 2 1
Saída
5.00

Entrada
4 1 1 5 2 -1 3 4 2 -1 -3 4 2
Saída
2.00 4.00 7.00

Entrada
4 15.89 0.7 0.53 0.45 0.38 0.22 0 0 0 0 0 1
Saída
15.44 0.45 1.00

18 Sentença Dançante

 (+++)

Uma sentença é chamada de dançante se sua primeira letra for maiúscula e cada letra subsequente for o oposto da letra anterior. Espaços devem ser ignorados ao determinar o case (minúsculo/maiúsculo) de uma letra. Por exemplo, "A b Cd"é uma sentença dançante porque a primeira letra (’A’) é maiúscula, a próxima letra (’b’) é minúscula, a próxima letra (’C’) é maiúscula, e a próxima letra (’d’) é minúscula.

Entrada

A entrada contém vários casos de teste. Cada caso de teste é composto por uma linha que contém uma sentença, que é uma string que contém entre 1 e 50 caracteres (’A’-’Z’,’a’-’z’ ou espaço ’ ’), inclusive, ou no mínimo uma letra (’A’-’Z’,’a’-’z’). A entrada termina por fim de arquivo.

Saída

Transforme a sentença de entrada em uma sentença dançante (conforme o exemplo abaixo) trocando as letras para minúscula ou maiúscula onde for necessário. Todos os espaços da sentença original deverão ser preservados, ou seja, "sentence "deverá ser convertido para "SeNtEnCe ".

Exemplo

Entrada

`This is a dancing sentence`
`  This is a dancing sentence`
`aaaaaaaaaaa`
`z`

Saída

`ThIs Is A dAnCiNg SeNtEnCe`
`  ThIs Is A dAnCiNg SeNtEnCe`
`AaAaAaAaAaA`
`Z`

19 *string to int* (+++)

 (+++)

Faça um programa que leia um número inteiro fornecido como uma *string* e o converta para um `long int`. Você deve implementar a função:

```
/**
 * Converte a string str para o valor inteiro correspondente.
 * @param str string contendo um número inteiro
 * @return o número inteiro correspondente
 */
long int string2int( const char * str );
```

Entrada

O programa deve ler uma sequência de *strings* contento um número inteiro, de no máximo 128 caracteres, usando o comando: `scanf("%s", str);`, até atingir o final do arquivo, ou seja, usando o laço: `while( scanf("%s", str) != EOF ) { ... }`.

Saída

A saída é composta por linhas contendo o número inteiro e o seu dobro impressos usando o comando `printf("%ld %ld\n", n, n*2);`, onde *n* é o número convertido.

Observações

Para interromper o programa no Terminal use o comando Ctrl+D.

Exemplo

Entrada	Saída
1 -2 3 -4	1 2 -2 -4 3 6 -4 -8

Entrada	Saída
15	15 30

Entrada	Saída
-1234	-1234 -2468

20 Apague e Ganhe (++++)

 (++++)

Juliano é fã do programa de auditório Apagando e Ganhando, um programa no qual os participantes são selecionados através de um sorteio e recebem prêmios em dinheiro por participarem. No programa, o apresentador escreve um número de n dígitos em uma lousa. O participante então deve apagar uma certa quantidade de dígitos do número que está na lousa; o número formado por exatamente d dígitos que restaram é então o prêmio do participante. Juliano finalmente foi selecionado para participar do programa, e pediu que você escrevesse um programa que, dados o número que o apresentador escreveu na lousa, e quantos dígitos d devem restar na lousa, determina o valor do maior prêmio que Juliano pode ganhar.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros n e d ($1 \leq d < n \leq 10^5$), indicando a quantidade de dígitos do número que o apresentador escreveu na lousa e quantos dígitos devem restar do número, após Juliano apagar alguns dígitos do número dado. A linha seguinte contém o número escrito pelo apresentador, que não contém zeros à esquerda. O final da entrada é indicado por uma linha que contém apenas dois zeros, separados por um espaço em branco.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha na saída, contendo o maior prêmio que Juliano pode ganhar.

Exemplo

Entrada
4 2 3759 6 3 123123 7 4 1000000 0 0
Saída
79 323 1000

21 Loteria (++++)

 (++++)

A Mega-Sena é a maior loteria do Brasil. Para ganhar o prêmio máximo é necessário acertar a sena, o que significa obter coincidência entre seis dos números apostados e os seis números sorteados, de um total de sessenta dezenas (de 01 a 60), independentemente da ordem da aposta ou da ordem do sorteio. O concurso prevê também a chance de ganhar parte do prêmio, acertando a quina ou a quadra. A Mega-Sena foi lançada em março de 1996 e já premiou mais de 200 ganhadores na faixa principal. Os prêmios correspondem a 32,2% da renda das apostas ao imposto de renda correspondem 13,8% de todas as apostas. Os vencedores têm 90 dias para retirar o prêmio, se o período expirar, o dinheiro do prêmio será transferido ao Tesouro Nacional e investido em programas educacionais. Vale lembrar que a probabilidade de acerto em uma única aposta de 6 dezenas é de 1 em 50.063.860, o que representa um percentual de 0,000002%. Faça um programa que receba todas as apostas e as seis dezenas sorteadas de um concurso e mostre quantos vencedores para sena, quina e quadra houve.

Entrada

Na primeira linha da entrada haverá uma linha com as seis dezenas sorteadas, separadas por um espaço em branco cada. Na linha seguinte haverá um inteiro N, $1 \leq N \leq 50000$, representando a quantidade de apostas. Em seguida, em cada uma das N linhas haverá as seis dezenas de cada aposta, sendo que as dezenas estão no intervalo entre 1 e 60 e sem repetição de dezenas por apostas.

Saída

A saída consiste de 3 linhas contando uma das seguintes frases: “Houve K acertador(es) da sena” ou “Houve K acertador(es) da quina” ou ainda “Houve K acertador(es) da quadra”, onde K é quantidade de acertadores para a faixa. Caso não haja acertadores a seguinte frase deve ser apresentada: “Nao houve acertador para sena” ou “Nao houve acertador para quina” ou ainda “Nao houve acertador para quadra”. Ao exibir a última frase quebre uma linha.

Exemplo

Entrada
23 12 33 19 10 8 5 23 19 8 12 60 18 14 60 12 44 54 10 8 3 12 19 33 10 33 15 7 60 12 10 22 12 19 23 33 11
Saída
Nao houve acertador para sena Houve 1 acertador(es) da quina Houve 2 acertador(es) da quadra

22 Os Verdadeiros Sete Anões da Branca de Neve (++++)

 (++++)

Todos os dias, enquanto os anões estão ocupados nas minas, Branca de Neve prepara o jantar para eles: sete cadeiras, sete pratos, sete garfos e sete facas para sete anões famintos. Um dia, em vez de sete, nove anões voltaram das minas (ninguém sabe como ou por quê). Cada um deles afirma ser um dos sete anões da Branca de Neve. Felizmente, cada anão usa uma touca com um número inteiro positivo (menor que 100) escrito nela. Branca de Neve, uma matemática famosa, já havia observado, há muito tempo, que a soma dos números nas toucas de seus sete anões era exatamente 100. Escreva um programa que determina quais anões são legítimos, ou seja, escolhe sete dos nove números que totalizem 100.

Entrada

A entrada conterá um inteiro T , o número de casos de testes, e, para cada caso de teste, nove linhas de entrada. Cada uma com um inteiro entre 1 e 99 (inclusive). Todos os números serão distintos.

Saída

A saída deve conter, para cada caso de teste, exatamente sete linhas. Cada uma com um dos números nas toucas dos anões de Branca de Neve (em ordem crescente).

Exemplo

Entrada
2 7 8 10 13 15 19 20 23 25 8 6 5 1 37 30 28 22 36
Saída
7 8 10 13 19 20 23 1 5 6 8 22 28 30

23 Intercala (+++++)

 (+++++)

Faça um algoritmo que aloque dois vetores V1 e V2 com o tamanho de cada entrada q1 e q2, receba os q1 valores no vetor V1 e os q2 valores no vetor V e construa um terceiro vetor, Vr, com a intercalação dos vetores V1 e V2 de forma ordenada.

Entrada

A entrada consiste de dois número positivo q1 e q2 , sendo $0 < q(1,2) \leq 500000$, representando a quantidade de entradas do programa. Seguido de q1 +q2 linhas, onde nas q1 primeiras linhas estão os q1 valores e nas demais q2 linhas estão os q2 valores. Esses valores são naturais n, $0 \leq n \leq 999999$. E ainda, dentro do mesmo bloco é garantido que o número n representado na linha q é menor que o número que está em q+1 e maior que ou igual ao que está em q-1. Ou seja: $n(q-1) \leq n(q) < n(q+1)$ para todo q.

Saída

A saída deverá ser todos os q1 +q2 valores das duas entradas intercalados e impressos de forma crescente.

Exemplo

Entrada
5 7 1 3 5 7 21 0 2 4 6 8 10 12
Saída
0 1 2 3 4 5 6 7 8 10 12 21