Lista-Vetores e Strings

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 Elementos Pares do Vetor (+)

 (+)

Faça um programa que receba um vetor V de N inteiros e imprima os elementos pares do vetor V.

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 gera apenas uma linha contendo os elementos pares do vetor V, separados por um espaço em branco cada. Após o último número par impresso, mostre a quantidade de números pares presentes no vetor V. Após a impressão do último valor quebre uma linha.

Exemplo

Entrada
5 7 8 4 9 2
Saída
8 4 2 3

Entrada
4 3 3 3 3
Saída
0

Entrada
8 235 6 23 5 78 123 89 4
Saída
6 78 4 3

Entrada
1 7
Saída
0

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

Entrada
10 2 8 6 4 20 18 34 0 8 10
Saída
2 8 6 4 20 18 34 0 8 10 10

4 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

5 Ler e Imprimir (+)

 (+)

Escreva um programa em C para ler n elementos inteiros em um vetor, e depois imprimi-los.

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 valores lidos.

Exemplo

Entrada
5 1 2 3 4 5
Saída
1 2 3 4 5

6 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`

7 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

8 Zero Vale Zero (+)

 (+)

Um dia o Prof. Humberto José Roberto fez o seguinte questionamento: Se o zero a esquerda de um número não tem valor algum, por que teria em outras posições de um número? Analisando da seguinte forma, ele pede sua ajuda para, ao somar dois valores inteiros, que o resultado seja exibido segundo o raciocínio dele, ou seja, sem os Zeros. Por exemplo, ao somar 15 + 5, o resultado seria 20, mas com esta nova ideia, o novo resultado seria 2, e, ao somar 99 + 6, o resultado seria 105, mas com esta nova ideia, o novo resultado seria 15.

Escreva um programa que, dado dois números inteiros, sem o algarismo zero, some os mesmos e, caso o resultado tenha algum algarismo zero, que os retire antes de exibir.

Entrada

Haverá diversos casos de teste. Cada caso de teste inicia com dois inteiros M e N $(1 \leq M \leq N \leq 999.999.999)$. O último caso de teste é indicado quando $N = M = 0$, sendo que este caso não deve ser processado.

Saída

Para cada caso de teste, imprima o resultado da soma dos dois valores, sem os zeros.

Sugestão

Ao somar os dois números utilize a função sprintf() para armazenar a soma em uma string.

Exemplo

Entrada
7 8 15 5 99 6 0 0
Saída
15 2 15

9 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

10 Conversão de Decimal para Binário (++)

 (++)

Escreva um programa que leia vários números inteiros positivos na base decimal e escreva os valores correspondentes desses números na base binária.

Entrada

A entrada contém várias linhas, cada uma contendo um valor inteiro N na base decimal tal que seu número binário equivalente possui no máximo 32 bits. A saída termina por fim de arquivo.

Saída

A saída é formada por tantas linhas quantas forem as linhas na entrada. Cada linha contém a conversão para a base binária do valor em decimal que aparece na linha correspondente da entrada.

Exemplo

Entrada
0 1 2 3 5 572
Saída
0 1 10 11 101 1000111100

11 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

12 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

13 Ordena Lista (++)

 (++)

Faça um programa para imprimir uma lista de inteiros em ordem crescente.

Entrada

A entrada contém apenas um caso de teste. Na primeira linha há um inteiro N, $1 < N \leq 1000$, representando a quantidade de inteiros que serão informados. Em seguida haverá N linhas com um inteiro em cada linha.

Saída

Seu programa gera N linhas de saída, contendo em cada linha um inteiro que são os mesmos informados na entrada, porém em ordem crescente. Após o último número impresso, quebre uma linha.

Exemplo

Entrada
5 7 1 3 4 5
Saída
1 3 4 5 7
Entrada
2 4 3
Saída
3 4

14 Maior Frequencia (++)

 (++)

Dada uma sequência de N números entre 0 e 100. Determine qual o valor de maior frequência. Caso haja mais de um valor tenha a maior frequência, mostre o menor deles.

Entrada

Na primeira linha há um inteiro N, $1 \leq N \leq 1000000$, representando a quantidade números. Nas N linhas seguintes haverá um número natural entre 0 e 100 inclusive por linha.

Saída

O programa gera apenas duas linhas. Na primeira dela mostre qual foi o valor com maior frequência. E na segunda linha, mostre a quantidade de vezes que esse número apareceu na sequência de valores. Após a impressão deste último valor quebre uma linha. Caso haja mais de um valor tenha a maior frequência, mostre o menor deles.

Exemplo

Entrada
10 1 7 4 29 7 4 7 8 7 29
Saída
7 4

15 Menor Distancia (++)

 (++)

Calcular a distância entre os dois elementos mais próximos em uma sequência de N números inteiros.

Entrada

Na primeira linha há um inteiro T, $1 \leq T \leq 10$, representando a quantidade de testes a serem realizados. Para cada teste são esperados: (a) um número inteiro N, $2 \leq N \leq 1000$, que é o tamanho do vetor, e (b) uma sequência de N números inteiros que são os elementos do vetor. No vetor cada elemento deve estar no intervalo [-1000 , 1000].

Saída

O programa apresenta, para cada teste, o distância entre os dois elementos mais próximos e o número de comparações realizadas para resolver o problema. Observe que o número de comparações requeridas para resolver o problema para um vetor de tamanho N é N^2.

Exemplo

Entrada
4 6 1 2 3 4 5 6 10 1 1 1 1 1 1 1 1 1 1 4 20 40 21 100 6 10 20 30 40 50 55
Saída
1 36 0 100 1 16 5 36

Entrada
3 20 0 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 20 90 80 70 60 50 40 30 20 10 0 1 100 200 300 400 500 600 700 800 900 20 90 80 70 60 50 40 30 20 10 0 100 200 300 400 500 600 700 800 900 1000
Saída
0 400 1 400 10 400

Entrada
1 2 0 1000
Saída
1000 4

Entrada
1 2 -1000 1000
Saída
2000 4

16 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

17 Procura Caractere (++)

 (++)

Escreva um programa para ler várias linhas na entrada. Cada linha contém um caractere seguido por um espaço e por uma string. Para cada linha, o programa deve chamar uma função do tipo int que receba como primeiro parâmetro o caractere lido e como segundo parâmetro a string lida. A função deve retorna o índice do vetor onde o caractere aparece pela primeira vez na string. Se o caractere não aparece na string, a função deve retornar -1.

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 caractere, um espaço e uma string, com no máximo 499 caracteres.

Saída

Para cada caso de teste o programa deve imprimir uma das seguintes frases:

- "Caractere *c* encontrado no indice *i* da string.", ou
- "Caractere *c* nao encontrado."

Exemplo

Entrada:
4 o Introducao a Programacao G Universidade Federal de Goias ; Universidade Federal de Goias   Universidade Federal de Goias

Saída:
Caractere o encontrado no indice 4 da string. Caractere G encontrado no indice 24 da string. Caractere ; nao encontrado. Caractere encontrado no indice 12 da string.

Observação: Na última linha de entrada do exemplo, o caractere a ser procurado é o caractere espaço.

18 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

19 Aliteração (++)

 (+++)

Uma aliteração ocorre quando duas ou mais palavras consecutivas de um texto possuem a mesma letra inicial (ignorando maiúsculas e minúsculas). Sua tarefa é desenvolver um programa que identifique, a partir de uma sequência de palavras, o número de aliterações que essa sequência possui.

Entrada

A entrada contém diversos casos de testes. Cada caso é expresso como um texto em uma única linha, contendo de 1 a 100 palavras separadas por um único espaço, cada palavra tendo de 1 a 50 letras minúsculas ou maiúsculas ('A'-'Z','a'-'z'). A entrada termina em EOF.

Saída

Para cada caso de teste imprima o número de aliterações existentes no texto informado, conforme exemplos abaixo.

Exemplo

Entrada

`He has four fanatic fantastic fans`
`There may be no alliteration in a sequence`
`Round the rugged rock the ragged rascal ran`
`area artic Soul Silly subway ant artic none`

Saída

`2`
`0`
`2`
`3`

20 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

21 Avance as Letras (+++)

 (+++)

São dadas na entrada uma string A e outra B. Em uma operação você pode escolher uma letra da primeira string e avançar esta letra. Avançar uma letra significa transformá-la na próxima letra do alfabeto, veja que a próxima letra depois de z vem a letra a novamente!

Por exemplo, podemos transformar a string **ab** em **bd** em no mínimo 3 operações: **ab** $\rightarrow$ **bb** $\rightarrow$ **bc** $\rightarrow$ **bd**. Podemos aplicar operações nas letras em qualquer ordem, outra possibilidade seria: **ab** $\rightarrow$ **ac** $\rightarrow$ **bc** $\rightarrow$ **bd**.

Dadas as duas strings, calcule o mínimo número de operações necessárias para transformar a primeira na segunda.

Entrada

Na primeira linha terá um inteiro $T(T \leq 100)$ indicando o número de casos de teste. Para cada caso, na única linha teremos as duas strings $A(1 \leq |A| \leq 10^4$ - sendo que $|A|$ significa o tamanho da string $A)$ e $B(|B| = |A|)$ separadas por um espaço. Ambas as strings são compostas apenas por letras minúsculas do alfabeto e são do mesmo tamanho.

Saída

Para cada caso imprima o número mínimo de operações.

Exemplo

Entrada
3 ab bd abc abc abcdefghiz aaaaaaaaaa
Saída
3 0 173

22 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

23 CPF (+++)

 (+++)

Você foi contratado pelas Indústrias Udilandenses (INUDIL) para desenvolver uma maneira de verificar se o Cadastro de Pessoa Física (CPF) indicado por um cliente era válido ou não. Conversando com amigos, você chegou à conclusão de que um CPF seria válido se a soma de todos os seus dígitos resultasse em número múltiplo de 11. Após verificação minuciosa, você descobriu que essa maneira só funciona em cerca de 80% dos casos, e você precisa de mais do que isso para garantir a qualidade do seu trabalho. Após pesquisar mais, você descobriu que dos 11 dígitos do CPF, os dois últimos são verificadores e dependem dos 9 dígitos anteriores. Vamos introduzir alguma notação. Considere um CPF com os seguintes dígitos

$$a_1a_2a_3a_4a_5a_6a_7a_8a_9 - b_1b_2$$

Para descobrirmos o dígito b_1, procedemos da seguinte maneira: multiplicamos o primeiro dígito por 1, o segundo por 2, o terceiro por 3, o quarto por 4 e vamos assim até multiplicarmos o nono por 9. Então, somamos tudo isto. Após termos somado tudo, dividimos por 11. O dígito b_1 será o resto da divisão (ou 0, caso o resto seja 10).

Para o segundo dígito verificador, temos o seguinte: multiplicamos o primeiro elemento por 9, o segundo por 8, o terceiro por 7, o quarto por 6 e vamos assim até multiplicarmos o nono por 1. Então, somamos tudo isto e dividimos por 11. O dígito b_2 será o resto da divisão (ou 0, caso o resto seja 10).

Sabendo que isso vale para 100% dos CPFs, sua missão é implementar um programa que, dado um CPF, diga se ele é válido ou não.

Entrada

A entrada conterá uma linha com um inteiro T, que indica o número de casos de testes. Esta linha é seguida por T linhas, cada uma contendo uma sequência de 11 dígitos decimais, separados entre sis por um espaço. Após o último dígito decimal segue o caractere de quebra de linha.

Saída

Para cada candidato a CPF da entrada, escreva "CPF valido", se ele for um CPF válido e, escreva "CPF invalido", em caso contrário.

Exemplos

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

Saída
CPF invalido CPF valido CPF invalido CPF valido CPF invalido

24 Elementos Únicos (+++)

 (+++)

Dada uma sequência de N números inteiros na ordem crescente, identifique os elementos únicos.

Entrada

Na primeira linha há um inteiro N, $1 \leq N \leq 1000$, representando a quantidade de elementos. Nas N linhas seguintes haverá um número inteiro (positivo ou negativo) por linha.

Saída

O programa apresenta um sequência de elementos únicos, na ordem crescente. Cada elemento é apresentado em uma linha.

Exemplo

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

Entrada
5 9 9 9 9 9
Saída
9

Entrada
7 4 4 4 4 4 4 8
Saída
4 8

Entrada
1 900
Saída
900

Entrada
3 -200 -200 -10
Saída
-200 -10

25 Frequência de Letras (+++)

 (+++)

Neste problema estamos interessados na frequência das letras em uma dada linha de texto. Especificamente, deseja-se saber qual(is) a(s) letra(s) de maior frequência do texto, ignorando o "case sensitive", ou seja maiúsculas ou minúsculas (sendo mais claro, "letras" referem-se precisamente às 26 letras do alfabeto).

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 caracteres "não letras", mas é garantido que tenha ao menos uma letra e que tenha no máximo 200 caracteres no total.

Saída

Para cada caso de teste, imprima uma linha contendo a(s) letra(s) que mais ocorreu(ocorreram) no texto em minúsculas (se houver empate, imprima as letras em ordem alfabética).

Exemplo

Entrada
3 Computers account for only 5% of the country's commercial electricity consumption. Input frequency letters
Saída
co inptu e

26 Limpa *String* (+++)

 (+++)

Faça um programa que atualize um texto removendo uma lista de caracteres indesejados. Tanto o texto quanto a lista de caracteres devem ser lidos no formato de *strings*.

Escreva a função `str_clean` que realiza o processamento desejado. Ela deve receber como parâmetros a *string* original `str` e a *string* com caracteres indesejados `clr`. Considere o tamanho máximo de 256 caracteres.

Sua função `str_clean` deve varrer a *string* original e remover todos os caracteres que ocorrem na *string* `clr`. Use um vetor de no máximo 256 caracteres. Seu programa principal deve ser o seguinte código:

```
int main() {
    char str[N]; // string original
    char clr[N]; // lista de caracteres indesejados
    scanf("%[^\n]", str);
    scanf("\n%[^\n]", clr);
    str_clean(str, clr) ;
    printf("%s\n", str);
    return 0;
}
```

Entrada

Seu programa deve ler duas *strings*.

Saída

Uma linha contendo a *string* modificada.

Observações

Exemplo

Entrada	Saída
`Fulando de Tal da Silva` `aeiou`	`Flnd d Tl d Slv`

Entrada	Saída
`100 200 300 400 500 600 700` `123456789`	`00 00 00 00 00 00 00`

Entrada	Saída
`1111111111x` `1`	`x`

27 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

28 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`
`aaaaaaaaaa`
`z`

Saída

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

29 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

30 Comparação de textos (++++)

 (++++)

Um sistema inteligente de reconhecimento de textos precisa de um algoritmo que seja capaz de comparar frases. Você, um excelente projetista de sistemas de reconhecimento de padrões, sugeriu o seguinte método de comparação: dados duas *strings* A e B, a distância entre as A e B pode ser calculada usando a distância euclidiana entre os vetores de frequência das vogais que compõem cada *string*. Um vetor de frequências das vogais "a, e, i, o, u" , ou suas maiúsculas, é um vetor com 5 posições, onde cada posição armazena a quantidade de vezes que as vogais aparecem na *string*. Por exemplo:

Seja A ="ola, meu nome e maria", possui um vetor de frequências de vogais $F_A = (3,3,1,2,1)$, ou seja, há 3 vogais "a", 3 vogais "e", 1 vogal "i", "2 vogais "o"e 1 vogal "u".

Para a *string* B ="era uma vez um lobo mal...", $F_B = (3,2,0,2,2)$.

A distância entre A e B é dada pela equação:

$$d(A,B) = \sqrt{\sum_{i=0}^{4}(F_A(i) - F_B(i))^2} \tag{1}$$

onde, $F_A(i)$ e $F_B(i)$ é a quantidade de vezes que a vogal i aparece nas *strings* A e B respectivamente.

Para o exemplo dado, o resultado da distância seria:

$$d(A,B) = \sqrt{(3-3)^2 + (3-2)^2 + (1-0)^2 + (2-2)^2 + (1-2)^2} = 1.732050808 \tag{2}$$

Faça um programa que leia duas *strings*, calcule e apresente a distância entre elas usando o método descrito.

Entrada

O programa deve ler uma linha contendo 2 *strings*, cada uma de no máximo 1000 caracteres, separadas pelo caracter ';'.

Saída

Se o texto informado não conter o caracter separador ';' ou mais de um caracter ';', o programa deve imprimir a mensagem "FORMATO INVALIDO!". Caso contrário, o programa deve apresentar 3 linhas. As duas primeiras devem conter os vetores de frequências de cada *string*, com os valores entre parênteses e separados por vírgulas, e a última linha deve conter o valor da distância entre as *strings* com 2 casas decimais.

Observações

O programa não deve diferencias maiúsculas de minúsculas. Também não não são admitidos acentos no texto de entrada.

Exemplo

Entrada
`Ola mundo, meu nome e Maria; Era uma vez um lobo mal...`
Saída

`(3,3,1,3,2)`
`(3,2,0,2,2)`
`1.73`

Entrada
`Eu serei um grande Cientista da Computacao.`
Saída
`FORMATO INVALIDO!`

31 Counting Sort (++++)

 (++++)

Counting sort é um algoritmo de ordenação estável cuja complexidade é O(n). As chaves podem tomar valores entre 0 e M-1. Se existirem k_0 chaves com valor 0, então ocupam as primeiras k_0 posições do vetor final: de 0 a k_{0-1}.

O procedimento para implementação do Counting Sort segue os seguintes passos:

1. Cria-se um vetor vCount[M+1] e vOrd[N-1], onde N é a quantidade de elementos a serem ordenador e M é o maior valor entre os elementos a serem ordenados.
2. Inicializa-se todas as posições de vCount com 0.
3. Percorre-se o vetor v e, para cada posição i de v faz-se vCount[v[i]]++, o que faz com que, no final, cada posição i de vCount contem a quantidade de vezes que a chave i aparece em V.
4. Acumula-se em cada elemento de vCount o elemento somado ao elemento anterior, desta forma, vCount[i] indica a posição-1 ordenada do primeiro elemento de chave i.
5. Guarda-se em vOrd os valores de V ordenados de acordo com vOrd[vCount[v[i]-1]=V[i]. E decrementa-se vCount[v[i]] de uma unidade.
6. Copia-se vOrd para v.

Esta implementação tem a desvantagem de precisar de vetores auxiliares. O Counting Sort ordena exclusivamente números inteiros pelo fato de seus valores servirem como índices no vetor de contagem.

Entrada

O programa possui vários casos de testes. A primeira de cada caso contem um inteiro N, $1 < N \leq 10000$, representando o tamanho do vetor. A segunda linha conterá N inteiros entre 0 e 1000, representando os N elementos do vetor. A entrada termina quando N=0.

Saída

O programa gera uma linha de saída para cada entrada, contendo os valores recebidos na entrada ordenados de acordo com o Counting Sort. Entre cada valor há um espaço em branco. Antes do primeiro valor não deve-se imprimir nada e após ao último valor deve-se apenas quebrar uma linha.

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
3 3 6 6 7 9 13 13 14 14 5 6 7 8 9 0 1 2 3 4 5 6 7

32 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

33 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

34 Counting Sort (++++)

 (++++)

Counting sort é um algoritmo de ordenação estável cuja complexidade é O(n). As chaves podem tomar valores entre 0 e M-1. Se existirem k_0 chaves com valor 0, então ocupam as primeiras k_0 posições do vetor final: de 0 a k_{0-1}.

O procedimento para implementação do Counting Sort segue os seguintes passos:

1. Cria-se um vetor vCount[M+1] e vOrd[N-1], onde N é a quantidade de elementos a serem ordenador e M é o maior valor entre os elementos a serem ordenados.
2. Inicializa-se todas as posições de vCount com 0.
3. Percorre-se o vetor v e, para cada posição i de v faz-se vCount[v[i]]++, o que faz com que, no final, cada posição i de vCount contem a quantidade de vezes que a chave i aparece em V.
4. Acumula-se em cada elemento de vCount o elemento somado ao elemento anterior, desta forma, vCount[i] indica a posição-1 ordenada do primeiro elemento de chave i.
5. Guarda-se em vOrd os valores de V ordenados de acordo com vOrd[vCount[v[i]-1]=V[i]. E decrementa-se vCount[v[i]] de uma unidade.
6. Copia-se vOrd para v.

Esta implementação tem a desvantagem de precisar de vetores auxiliares. O Counting Sort ordena exclusivamente números inteiros pelo fato de seus valores servirem como índices no vetor de contagem.

Entrada

O programa possui vários casos de testes. A primeira de cada caso contem um inteiro N, $1 < N \leq 10000$, representando o tamanho do vetor. A segunda linha conterá N inteiros entre 0 e 1000, representando os N elementos do vetor. A entrada termina quando N=0.

Saída

O programa gera uma linha de saída para cada entrada, contendo os valores recebidos na entrada ordenados de acordo com o Counting Sort. Entre cada valor há um espaço em branco. Antes do primeiro valor não deve-se imprimir nada e após ao último valor deve-se apenas quebrar uma linha.

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
3 3 6 6 7 9 13 13 14 14 5 6 7 8 9 0 1 2 3 4 5 6 7

35 União e intersecção de conjuntos (++++)

 (++++)

Faça um programa que leia 2 conujuntos (A e B) válidos, sem elementos repetidos, cada um com no mínimo 1 e no máximo 100 elementos, e imprima $A \cup B$ e $A \cap B$.

Entrada

Durante a entrada de dados, o programa deve ler tamanhos válidos para os conjuntos A e B. Caso o tamanho informado seja inválido, o programa deve refazer a leitura do tamanho do conjunto. A leitura deve ser feita separadamente. Uma vez definido os tamanhos dos vetores, o programa deve ler os elementos de cada vetor. Em seguida, o programa deve apresentar na tela uma linha com o conjunto união, ou seja, todos os elementos que ocorrem em A e B, e outra linha contendo o conjunto intersecção, ou seja, todos os elementos que ocorrem em A e B.

O programa deve ler um número inteiro T_A, correspondente ao tamanho do conjunto A, até que T_A seja válido, em seguida outro número inteiro T_B, correspondente ao tamanho do conjunto B até que T_B seja válido. Uma vez definido os tamanhos dos vetores, o programa deve ler $T_A + T_B$ elementos, correspondentes aos elementos de A e B. Durante a leitura dos elementos de um conjunto, o programa deve permitir somente a leitura de elementos diferentes aos já presentes no conjunto. Caso um elemento lido já esteja presente no conjunto, o programa deve ignorá-lo e realizar uma nova leitura do elemento.

Saída

O programa deve apresentar na tela uma linha com o conjunto união, ou seja, todos os elementos que ocorrem em A e B, e outra linha contendo o conjunto intersecção, ou seja, todos os elementos que ocorrem em A e B. Os elementos dos conjuntos devem ser apresentados entre parênteses, separados por vírgula e sem espaços.

Observações

Não se esqueça que um conjunto válido não permite a existência de elementos repetidos.

Exemplo

Entrada	Saída
3 2 1 2 3 1 2	(1,2,3) (1,2)

Entrada	Saída
0 0 1001 2 -1 4 5 9 0 5 7 2	(5,9,0,7,2) (5)

Entrada	Saída
0 0 1001 2 -1 4 5 9 9 5 0 0 0 7	(5,9,0,7) (9,5)

Entrada	Saída
1 1 5 9	(5,9) ()

36 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