

Prova Segunda Unidade

Vetores

Bino e Cino são colegas inseparáveis. Bino gosta de criar desafios matemáticos para Cino resolver. Desta vez, Bino gerou uma lista de números e perguntou ao Cino quantos números da lista são múltiplos de 2,3,4,5 e 6.

Esse desafio pode parecer simples, porém, quando a lista contém muitos números, Cino se confunde e acaba errando alguns cálculos. Para ajudar Cino, faça um programa para resolver o desafio de Bino.

Entrada

A primeira linha da entrada consiste em um inteiro N ($1 \leq N \leq 1000$), representando a quantidade de números na lista de Bino. A segunda linha contém N inteiros L_i ($1 \leq L_i \leq 100$), representando os números da lista de Bino.

Saída

Imprima a quantidade de números múltiplos de 2,3,4,5 e 6 presentes na lista.

Exemplo de Entrada:	Exemplo de saída
5	4 Multiplo(s) de 2
2 5 4 20 10	0 Multiplo(s) de 3
	2 Multiplo(s) de 4
	3 Multiplo(s) de 5
	0 Multiplo(s) de 6

Obs: Para esta questão é obrigatório o uso de **vetores**. A utilização de **funções** é opcional.

Strings

O famoso imperador romano César usava um sistema de criptografia simples para se comunicar com seus generais. Neste sistema, cada letra deveria ser substituída pela letra seguinte no alfabeto. No esquema, a letra seguinte à última era a primeira do alfabeto.

Por exemplo, a mensagem:

Atacar hoje.

Após a codificação fica:

Bubdbbs ipkf.

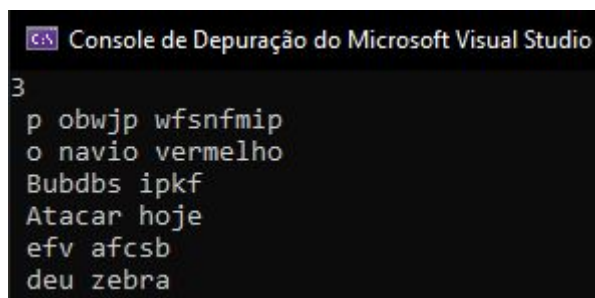
A sua tarefa é escrever um programa que **DECODIFIQUE** as mensagens que o imperador César envia para seus soldados.

Entrada

O programa deve começar lendo um inteiro N ($0 \leq N \leq 1000$) indicando quantas mensagens devem ser decodificadas. A seguir, o programa deve ler N mensagens, cada uma com até 80 caracteres escritos em única linha. Como caracteres, são aceitas letras maiúsculas e minúsculas e espaços.

Saída

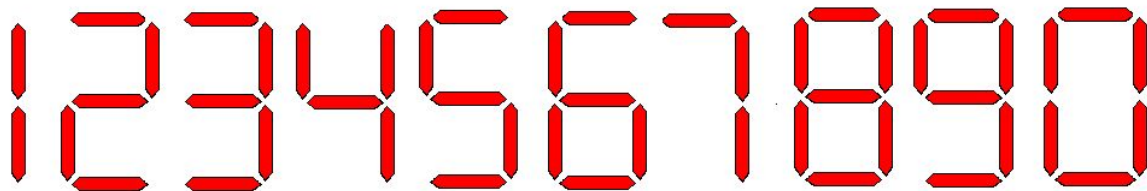
Para cada mensagem, seu programa deve imprimir a mensagem decodificada de acordo com a criptografia de César.



```
3
p obwj p wfsnfmip
o navio vermelho
Bubdbbs ipkf
Atacar hoje
efv afcsb
deu zebra
```

String + Funções

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.



No qual cada traço do display corresponde a um LED:

Por exemplo:

Se João deseja mostrar o número: 120 a quantidade de LEDs que ele irá necessitar é:

- Para o número 1 - Dois LEDs
- Para o número 2 - Cinco LEDs
- Para o número 0 - Seis LEDs,

Totalizando: 13 LEDs

Se João deseja mostrar o número **32**, a quantidade de LEDs que ele irá necessitar é:

Para o número **3** - Cinco LEDs

Para o número **2** - Cinco LEDs

Totalizando: 10 LEDs

```
Console de Depuração do Microsoft Visual Studio
Digite a quantidade de numeros: 4
Digite o numero que deseja descobrir a qtd de LEDs: 32
Serao necessarios: 10 leds
Digite o numero que deseja descobrir a qtd de LEDs: 1
Serao necessarios: 2 leds
Digite o numero que deseja descobrir a qtd de LEDs: 2568
Serao necessarios: 23 leds
Digite o numero que deseja descobrir a qtd de LEDs: 159753
Serao necessarios: 26 leds
```

Entrada

A entrada contém um inteiro N , ($1 \leq N \leq 1000$) correspondente ao número de casos de teste, seguido de N linhas, cada linha contendo um número ($1 \leq V \leq 100$) 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".

Atenção, utilize o protótipo abaixo. A função `contarLeds` deve retornar a quantidade de Leds necessários para escrever o número, sintá-se livre para acrescentar outras funções se achar necessário.

```
#include <iostream>
#include <cstring>
using namespace std;
int contarLeds() { //Alterar a declaração da função para receber o parâmetro
//Função para somar a quantidade de leds
}

int main() {
    //Comando para a leitura da String:
    cin >> leds;
    cout<< total_leds << " led ";
    return 0;
}
```

Vetores + Funções

Considerando a entrada de valores inteiros não negativos, ordene estes valores segundo o seguinte critério:

- Pares em ordem crescente;
- Impares em ordem decrescente

Entrada

A primeira linha de entrada contém um único inteiro positivo ($1 < N < 105$). Este é o número de linhas de entrada que vem logo a seguir. As próximas N linhas conterão, cada uma delas, um valor inteiro não negativo.

Saída

Apresente todos os valores lidos na entrada segundo a ordem apresentada acima. Cada número deve ser impresso em uma linha, conforme exemplo abaixo.

Exemplo de Entrada

```
10
4 34 32 543 3456 654 567 87 6789 98
```

Exemplo de Saída

```
4 32 34 98 654 3456 6789 567 543 87
```

Struct

Depois de resgatar a princesa Peach, Super Mário e ela resolveram tirar férias e viajar entre cidades do reino dos cogumelos. Porém, na etapa de planejamento, eles gostariam de saber quais cidades ficam mais longe e mais perto da cidade de onde eles vão partir. Para fazer estes cálculos, eles dispõem das coordenadas (x,y) da cidade de partida e das cidades nas quais eles vão viajar.

Entrada:

Primeiro, o programa deve fazer a leitura de um número inteiro N representando o número de cidades (incluindo a cidade de origem). Em seguida, serão lidos N pares de números representando as coordenadas de cada uma das cidades que o casal irá visitar. O primeiro par de coordenadas pertence à cidade na qual eles partirão (a origem).

Saída

A saída é composta por três linhas. A primeira linha mostra a cidade de partida da viagem. A segunda linha é composta da cidade mais próxima a cidade de origem e a terceira linha é composta da cidade mais distante da cidade de origem. (Caso haja mais de uma cidade mais próxima ou mais distante da origem, basta mostrar uma na saída).

Obrigatório o uso de structs para representar as cidades.

Obs: A utilização de structs não é opcional.

Obs²: Vocês definem os membros da struct que considerarem importantes.

Dica: A distância entre dois pontos cartesianos é dada por:

```
Console de Depuração do Microsoft Visual Studio
10
2 5
3 4
6 7
9 1
1 1
0 0
10 10
15 1
1 0
3 5
A cidade de origem: (2,5)
A cidade mais proxima: (3,5)
A cidade mais distante: (15,1)
```

```
Console de Depuração do Microsoft Visual Studio
4
0 0
3 1
5 4
1 1
A cidade de origem: (0,0)
A cidade mais proxima: (1,1)
A cidade mais distante: (5,4)
```