ATENÇÃO:	**a)** Todos os vetores utilizados têm que ser declarados na função ***main***. **b)** Não é permitido utilizar variáveis globais. O #**define** pode ser utilizado.

1) Faça um algoritmo para verificar se um número real lido pelo teclado encontra-se ou não em um vetor com 30 números reais (também lido pelo teclado). Crie e utilize duas **funções**: uma para preencher o vetor e outra para verificar se o número pertence ou não ao vetor. A impressão desta informação (se o número pertence ou não ao vetor) deve ser na função main.

2) Faça um algoritmo para ler (pelo teclado) um vetor com 15 elementos inteiros e depois inverter este mesmo vetor, **sem utilizar um vetor auxiliar**. Crie e utilize três **funções**: uma para preencher o vetor, outra para invertê-lo e a terceira para imprimi-lo após a inversão.

Obs.: O objetivo deste exercício não é imprimir o vetor em ordem inversa, mas sim colocar os elementos dentro do vetor em ordem inversa.

3) Considere um vetor com 40 números inteiros positivos gerados aleatoriamente de 1 a 100. Faça um algoritmo para verificar o número de vezes que um número inteiro positivo n lido pelo teclado aparece neste vetor. O programa também deve informar em quais posições (índices) do vetor o número aparece, caso ele pertença ao vetor. Crie e utilize duas **funções**: uma para preencher o vetor e outra para realizar a verificação.

Obs. 1: O seu programa deve verificar **primeiro** quantas vezes o número n aparece no vetor. **Depois**, se ele aparecer alguma vez no vetor, imprimir as posições que ele aparece. Se ele não pertencer ao vetor, seu programa deve imprimir: "Número não pertence ao vetor".

Obs. 2: O exemplo da Figura 1 preenche e imprime um vetor com 10 números inteiros positivos de 1 até 10 gerados aleatoriamente.

```c
#include <stdio.h>

int main( ){

    int i, vetor[10];

    srand(time(NULL));  // Gera uma semente aleatória

    for(i = 0;  i < 10;  i++){
        vetor[i] = rand( ) % 10 + 1; // Gera números aleatórios de 1 a 10
        printf("%d ", vetor[i]);
    }

    return 0;

}
```

Figura 1: Exemplo de uso das funções rand e srand.

4) Faça um algoritmo para preencher um vetor (de tamanho 30) com elementos gerados aleatoriamente de 1 a 20, de maneira que não sejam inseridos números iguais no vetor. Ou seja, todos os números contidos no vetor têm que ser distintos. Crie e utilize duas **funções**: uma para preencher o vetor e outra para imprimi-lo.

5) Faça um algoritmo que leia pelo teclado os 30 números de um vetor do tipo float e imprima na tela o maior e o menor elementos deste vetor e a posição em que eles se encontram. Crie e utilize duas **funções**: uma para preencher o vetor e outra para imprimir as informações.

Obs.: Caso o maior e o menor elementos apareçam mais de uma vez no vetor, a posição a ser impressa é a do último maior e menor elementos.

6) Faça um algoritmo para ler uma palavra pelo teclado (com no máximo 15 caracteres) e verificar se ela é **Palíndrome** ou não, **sem utilizar qualquer estrutura de dados auxiliar**. Crie e utilize uma **função** para ler a palavra e uma **função** para fazer a verificação. A impressão da mensagem (se a palavra é ou não Palíndrome) deve ser na função *main*.

Obs. 1: Uma palavra Palíndrome é aquela que lida de frente para trás e de trás para frente tem a mesma sequência de caracteres. Exemplos: arara, ovo, asa, radar, matam, etc;

Obs. 2: Lembre-se que o usuário pode digitar letras maiúsculas e minúsculas.

7) Considere um vetor com 100 números lidos pelo teclado. Faça um algoritmo para ler pelo teclado (na função *main*) um número inteiro n ($1 \leq n \leq 100$) e imprimir todos os números que aparecem no vetor exatamente n vezes. Caso nenhum número apareça exatamente n vezes no vetor, esta informação deve ser impressa. Crie e utilize duas **funções**: uma para preencher o vetor e outra para imprimir o que se pede. Exemplo com um vetor de 10 números e $n = 2$:

Vetor: {3, 1, 9, 8, 3, 10, 1, 5, 9, 1}
Números que aparecem exatamente 2 vezes: 3, 9

Obs.: Não é permitido ordenar o vetor.

8) Considere um vetor com 50 números inteiros lidos pelo teclado. Faça um algoritmo que imprima o segundo maior número presente no vetor. Crie e utilize duas **funções**: uma para preencher o vetor e outro para imprimir o segundo maior número.

Obs. 1: Não é permitido utilizar qualquer estrutura de dados auxiliar.

Obs. 2: Não é permitido ordenar o vetor.

9) Considere um vetor com 50 números inteiros gerados aleatoriamente de 1 a 200. Faça um algoritmo para imprimir na tela a quantidade de **números compostos** presentes neste vetor. Crie e utilize três **funções**: uma para preencher o vetor, outra para imprimir a quantidade de números compostos e a terceira para retornar 1, se um número for composto, ou retornar 0, caso contrário.

Obs.: Um número natural é **Composto** quando ele tem mais do que dois divisores naturais distintos.

Exemplo: O número 6 é composto, pois tem como divisores os números 1, 2, 3 e 6.

10) Considere um vetor com 100 números inteiros gerados aleatoriamente de 1 a 500. Faça um algoritmo para imprimir na tela a quantidade de **números perfeitos** presentes neste vetor. Crie e utilize três **funções**: uma para preencher o vetor, outra para imprimir a quantidade de números perfeitos e a terceira para retornar 1, se um número for perfeito, ou retornar 0, caso contrário.

Obs.: Um número inteiro positivo é chamado de **Perfeito** se a soma dos seus divisores naturais distintos for igual ao próprio número, sem considerar o número como o seu próprio divisor.

Ex.: O número 6 é chamado de P, pois a soma dos seus divisores (1, 2 e 3) é igual a ele mesmo (1+2+3=6).

IMPORTANTE

A função para transformar uma letra para maiúscula é a **toupper** e para minúscula é a **tolower**, ambas da biblioteca **ctype.h**. Um exemplo da utilização destas funções pode ser visto na Figura 2.

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void lerFrase(char *frase);
void imprimeFrase(char *frase);

int main(){

    char frase[31];

    lerFrase(frase);

    imprimeFrase(frase);

    printf("\n\n");

    return 0;
}

void lerFrase(char *frase){

    printf("\nDigite a frase: ");
    scanf("%[^\n]s", frase); //Leitura da frase podendo conter espaços
}

void imprimeFrase(char *frase){

    int i;

    char letra;

    printf("\nFrase em maiusculo: ");
    for(i = 0; i < strlen(frase); i++){
        letra = toupper(frase[i]); //Transforma cada letra para maiúscula
        printf("%c", letra);
    }

    printf("\n\nFrase em minusculo: ");
    for(i = 0; i < strlen(frase); i++){
        letra = tolower(frase[i]); //Transforma cada letra para minúscula
        printf("%c", letra);
    }

}
```

Figura 2: Exemplo de uso das funções toupper e tolower.