Lista Vetores e Matrizes
Exercícios resolvidos

1. Faça uma função que receba um vetor e o inverta.

   Uma abordagem inicial seria usar um vetor auxiliar, no qual este receberia os valores do vetor original do final para o começo e depois o vetor original receberia uma cópia do vetor auxiliar. Apesar dessa abordagem resolver o problema, há o inconveniente da duplicação de espaço no caso de um vetor com muitos elementos. Na solução proposta a inversão é feita usando apenas o vetor original, e o percorrimento ocorre somente até a metade do vetor.

```
void inverteVetor(int vetor[], int tam){
    int i, aux;
    //(tam/2) pois so necessita percorrer até a metade
       do vetor
    for (i = 0; i < (tam/2); i++){
        aux = vetor[i];
        vetor[i] = vetor[tam - 1 - i];
        vetor[tam - 1 - i] = aux;
    }
}
```

2. Faça uma função que ordene os valores dentro de um vetor de tamanho n.

   A resolução desse problema não é trivial. Por outro lado, já existe uma série de algoritmos conhecidos para resolvê-lo. O algoritmo a seguir apresenta o algoritmo clássico de ordenação chamado Bolha (Bubblesort), que é um dos mais simples, porém menos eficiente.

```
void bubblesort(int vetor[], int tam){
    int aux;
    for (int i = tam; i > 0; i--){
        for (int j = 0; j < i; j++){
            if (vetor[j] > vetor[j+1]){
                aux = vetor[j];
                vetor[j] = vetor[j+1];
                vetor[j+1] = aux;
            }
        }
    }
}
```

3. Faça uma função que implemente o algoritmo de busca binária (busca de um determinado elemento em um vetor ordenado). Caso o valor seja encontrado, retorne a sua posição, caso contrário, retorne -1.

Uma das vantagens de se ter um vetor ordenado, usando algum algoritmo de ordenação, como o da questão anterior, é que há algoritmos de busca mais eficientes que o sequencial, como o algoritmo de busca binária.

```
int buscaBinaria(int vetor[], int tam, int elem){
    int i, lim_inf = 0, lim_sup = tam;
    do{
        i = (lim_inf + lim_sup)/2;
        if (vetor[i] < elem){
            lim_inf = i + 1;
        }else{
            lim_sup = i - 1;
        }
    } while(vetor[i] != elem || lim_inf < lim_sup);
    if(vetor[i] == elem){
        return i;
    }else{
        return -1;
    }
}
```

A busca binária começa a busca pelo meio do vetor, se o elemento buscado for maior que o elemento do meio, então a busca ocorre apenas na segunda metade do vetor, caso contrário, a busca ocorre apenas na primeira metade do vetor. Isso só funciona se o vetor estiver ordenado (em ordem crescente).

4. Dado uma matriz 4x4 de números inteiros, construa as seguintes funções para:

   a) Preencher a matriz com números aleatórios entre 0 a 9;

```
#define D 4
void sorteio(int M[][D]){
    int i, j;
    srand(time(NULL));
    for(i=0; i<D; i++)
        for(j=0; j<D; j++)
            M[i][j]= rand()%10;
}
```

   b) Imprimir a matriz;

```
void imprimir(int M[][D]){
    int i, j;
    for(i=0; i<D; i++){
        for(j=0; j<D; j++)
        printf("%d ", M[i][j]);
        printf("\n");
```

```
    }
}
```

c) Trocar as linhas da matriz;

```
//trocar duas linhas passadas como parametro da
   matriz
void trocarLinhas(int l1, int l2, int M[][D]){
    int j, aux;
    for(j=0; j<D; j++){
        aux = M[l1][j];
        M[l1][j] = M[l2][j];
        M[l2][j] = aux;
    }
}
```

d) Trocar as colunas da matriz;

```
//trocar duas colunas passadas como parametro da
   matriz
void trocarColunas(int c1, int c2, int M[][D]){
    int i, aux;
    for(i=0; i<D; i++){
        aux = M[i][c1];
        M[i][c1] = M[i][c2];
        M[i][c2] = aux;
    }
}
```

e) Trocar as diagonais (principal pela secundária);

```
void trocarDiagonais(int M[][D]){
    int i,c, aux;
    c = D;
    for(i=0; i<D; i++){
        c--;
        aux = M[i][i];
        M[i][i] = M[i][c];
        M[i][c] = aux;
    }
}
```

f) Construir a função main para uso das funções implementadas.

```
int main(){
    int op_menu, var1, var2;
    int matriz[D][D];
```

```
    sorteio(matriz);
    do{
        printf("Menu de opcoes\n");
        printf("1 - Imprimir Matriz\n");
        printf("2 - Trocar linhas\n");
        printf("3 - Trocar colunas\n");
        printf("4 - Trocar diagonais\n");
        printf("0 - sair\n");
        printf("Informe sua opcao: ");
        scanf("%d", &op_menu);
        switch(op_menu){
            case 1: imprimir(matriz);
            break;
            case 2:{
                printf("Informe a primeira linha: ");
                scanf("%d", &var1);
                printf("Informe a segunda linha: ");
                scanf("%d", &var2);
                if((var1>=0 && var1< D) &&
                (var2 >=0 && var2 < D)){
                    trocarLinhas(var1, var2, matriz);
                    printf("Linhas trocadas\n");
                }
                else
                printf("valor invalido para linhas\n"
                   );
                break;
            }
            case 3:{
                printf("Informe a primeira coluna: ")
                   ;
                scanf("%d", &var1);
                printf("Informe a segunda coluna: ");
                scanf("%d", &var2);
                if((var1>=0 && var1< D) &&
                (var2 >=0 && var2 < D)){
                    trocarColunas(var1, var2, matriz)
                       ;
                    printf("Colunas trocadas\n");
                }
                else
                printf("valor invalido para colunas\n
                   ");
                break;
```

```
            }
            case 4: trocarDiagonais(matriz);
            break;
            case 0:
            printf("sair"); break;
        }
    }while(op_menu != 0);

}
```