Lista de Exercícios

1. Dado o vetor formado pelos de valores {8, 5, 12, 3, 1, 9, 2, 15, 4, 13}, mostre o passo da execução dos seguintes algoritmos de ordenação;
   a) Insertionsort (mostrando o vetor resultante após cada valor inserido)
   b) Selectionsort (mostrando o vetor resultante após cada elemento "selecionado" e colocado em seu lugar);
   c) Mergesort (mostrando o vetor resultante após cada chamada do procedimento merge);
   d) Quicksort (mostrando o vetor resultante após cada chamada do procedimento partition);
   e) Heapsort (mostrando o heap máximo e o vetor resultante após a troca do primeiro com o ultimo elemento e a chamada do procedimento maxheapfy);
   f) Shellsort, com gaps de 7, 4 e 1(mostrando o vetor resultate após a ordenação por cada gap).
2. Ordene o vetor formado pelos valores {3, 1, 5, 2, 4, 2, 1, 3, 5} utilizando o algortimo countingsort. Mostre o vetor de contagem após cada laço de repetição existente no algoritmo (exceto o laço que inicializa tal vetor)
3. Ordene o vetor formado pelos valores {3, 919, 100, 5, 98, 0, 191, 12, 93, 8} utilizando o algoritmo radixsort. Mostre o vetor resultante após cada "subordenação"
4. Faça a inserção dos elementos com chave 24, 4, 10, 25, 8, 23, 9, 1, 12, na ordem apresentada, em uma tabela hash, cujo a função hash é k % 7 tratando as colisões com uma lista encadeada.
5. Altere o algoritmo countingsort para que ele possa ordenar valores com até 5 casas deciamais.
6. Altere o algorimo radixsort para que ele ordene um "vetor de strings", utilizando os códigos da tabela asc para comparar os elementos.
7. Uma maneira de melhorar o algoritmo quicksort é mistrurá-lo com outro algoritmo que tem tem o tempo de execução pior caso de $O(n \log_2 n)$. Dessa forma, é possível fazer um algoritmo que recebe a "profundidade" da chamada recursiva e um limite máximo de profundidade. Quando esse limite máximo de profundidade é alcançado, o algoritmo quicksort é substituido por um outro algoritmo de ordenação eficiente. Essa melhoria ao quicksort é chamada de Introsort. Considere a função a seguir;

```
void introsort(int v[], int n){
        int i,pmax=0;
        for(i=1;i<n;i*=2) pmax++; //calcular log2 n para profundidade máxima.
        intro_sort_rec(v,0,n-1,pmax,0);
}
```

   Implemente o procedimento intro_sort_rec, de maneira que quando a profundidade for menor que a profundiade máxima, o algoritmo introsort continue sendo chamado. Quando a profundiade alcançar a profundiade máxima, o algoritmo a ser executado seja o mergesort.
   Para a implementação do procedimento intro_sort_rec adote o seguinte formato;

```
void intro_sort_rec(int v[], int ini, int fim, int p_maxima, int p_atual);
```