1 - Explique detalhadamente o funcionamento do algoritmo de ordenação: **Insertion Sort.**

É um método bem simples de se pensar, pois ele procura o menor elemento de um vetor, e alinha ele para a esquerda do vetor. Possui complexidade C(n) = O(n) no melhor caso e O(n^2) no caso médio e pior caso.

2 - Escreva o código do algoritmo de ordenação **Insertion Sort** na linguagem de programação.

insertionSort.c > main()

```c
#include <stdio.h>

int main()
{
    int vet[10] = {4, 18, 6, 3, 8, 1, -4, 48, 0, -14}, i, j, aux;

    for(i = 1; i < 10; i++) {
        aux = vet[i];
        for(j = i-1; j >= 0 && aux < vet[j]; j--) {
            vet[j+1] = vet[j];
        }
            vet[j+1] = aux;
    }

    printf("\nVetor ordenado: ");

    for (i = 0; i < 10; i++) {
        printf(" %d", vet[i]);
    }

    return 0;
}
```

3 - Compare os pontos positivos e negativos dos algoritmos **Buble Sort**, **Selection Sort** e **Insertion Sort.**

O Selection Sort é o fácil de se fazer, geralmente quando precisamos ordenar algo em nosso cotidiano, inconscientemente pensamos em algo bem parecido com ele, porém a eficiência dele não é boa, mesmo nos melhores casos ele tem uma eficiência de C(n) = O(n^2).

O Buble Sort ou até mesmo o Insertion Sort, tem o mesmo nível de eficiência que o Selection Sort nos médios e piores casos, porem nos melhores casos a complexidade deles é C(n) = O(n).