SIN110 Algoritmos e Grafos exercício E01

1) A função abaixo recebe uma lista encadeada com cabeça e um inteiro x, e promete devolver p tal que p->chave==x ou NULL se tal p não existir. Analise a função verificando sua correção e eficiência.

```
  struct celula {
     int            chave;
     struct celula *prox;
  } ;

struct celula * busca (int x, struct celula * Y ){
   int achou = 0;
   struct celula * p = y->prox;
   while (p != NULL && !achou) {
      if (p->chave == x) achou = 1;
      p = p->prox;
      }
   if (achou) return p;
    else return NULL;
}
```

---

2) Prove a correção e determine o tempo de execução, no pior e no melhor caso, para o algoritmo de classificação abaixo:

```
ShakeSort(A, n)
1  e  ← 1
2  para i ← n-1 até e faça
3      para j ← e  até i faça
4          se A[j] > A[j+1]
5             então troca(A[j], A[j+1])
6      para j ← i  até e + 1 faça
7          se A[j-1] > A[j]
8             então troca(A[j-1], A[j])
9      e ← e + 1
```

---

3) Considere a seguinte algoritmo recursivo para calcular o máximo de um vetor ***A[e..d]***:

```
Max(A, e, d)
1  se e = d
2    então devolve A[e]
3    senão x ← ⌊(e+d)/2⌋
4          a ← Max(A, e ,x)
5          b ← Max(A, x+1, d)
6          se a > b
7              então devolve a
8              senão devolve b
```

Seja **C(n)** o número de vezes que a comparação da linha 6 é executada em uma chamada de ***Max(A,e,d)***, onde ***n = d – e + 1***. Escreva uma recorrência que define **C(n)** e, determine sua ordem de crescimento. Justifique sua resposta.

---

4) No método de classificação por inserção, uma otimização possível seria uma pesquisa mais rápida do local de inserção de uma chave através de busca binária. Escreva uma versão recursiva do algoritmo de ordenação por inserção implementando esta otimização considerando que o algoritmo deverá colocar em ordem crescente um vetor A[e..d]. Determine e compare a eficiência dos dois algoritmos para melhor e, para pior caso.
