Algoritmos e Estruturas de Dados I

1. Observe os algoritmos abaixo:

(a)
```
var N, r: Inteiro
ler(N); r ← 0
  enquanto (r+1)²≤N
    faça r ← r+1
escrever(r)
```

(b)
```
var N, r:
Inteiro ler(N);
r ← N
  enquanto r²>N
  faça r ← r-1
escrever(r)
```

(c)
```
var N, r, a, m: Inteiro
ler(N); r ← 0; a ← N+1
enquanto r+1 ≠ a faça m
     ← (r+a) div 2
    se m²≤N então
        r ← m
    senão
        a ← m
escrever(r)
```

   a. Descreva sucinta e precisamente o problema que tais algoritmos resolvem.
   b. Qual a complexidade de cada algoritmo?

2. Complete a função abaixo de modo que ela determine se o vetor B é um anagrama do vetor A, isto é, os elementos de B[1..n] consistem de uma permutação daqueles de A[1..n]. Em seguida, determine a complexidade de tempo da função elaborada.

```
função SãoAnagramas(A[]: Caractere, B[]: Caractere, N: Inteiro): Lógico
        //retornar verdadeiro se A[1..n] é anagrama de B[1..n]
      ?
```

3. Elabore os algoritmos abaixo para listas lineares **sequenciais não-ordenadas**. Determine a complexidade dos algoritmos elaborados. Considere a estrutura definida conforme abaixo:

Sequencial:
```
estrutura ListaLinear:
                E[1..1000]: Inteiro //elementos
                        N: Inteiro //número de elementos
```

   a. `procedimento Altera(ref L: ListaLinear, x: Inteiro, y: Inteiro):`
      // substitui o elemento x na lista linear sequencial L por y, mantendo o novo
      // elemento na mesma posição do antigo

   b. `função TodosDistintos(ref L: ListaLinear): Lógico`
      // retorna verdadeiro se os elementos de L são todos distintos

4. Elabore algoritmos para o exercício anterior, considerando agora que as listas são **encadeadas ordenadas**, com estrutura definida conforme abaixo:

Encadeada:
```
estrutura Nó:
                E: Inteiro //elemento
```

```
        Próx: ^Nó //próximo elemento
estrutura ListaLinear:
        Inicio: ^Nó //primeiro elemento
```