Nome: Flávio Eduardo Oliveira e Silva
Matrícula: 2017018013

Exercício 01 - Algoritmos Iterativos

1 - O algoritmo tem um ponto de parada, que é ao encontrar o resultado procurado, no qual a variável “achou” se torna diferente de 0, e quando não encontra o resultado procurado, em que o p se torna NULL, sendo realizado pela condicional (p != NULL && !achou), portanto, o algoritmo para.

Pior Caso

```
1   int achou = 0;                                  1
2   struct celula *p = y->prox;                     1
3   while (p != NULL && !achou( {                   n + 1
4       If (p->chave == X) achou == 1;              n
5       p = p->prox; }                              n
6   if (achou) return p;                            1
7   else return NULL;                               1

                                  Total: T(n) = 3n +5
```

A eficiência do algoritmo no pior caso é de **T(n) = 3n + 5** e seu comportamento assintótico é de **O(n) = n.**

Melhor Caso

```
1   int achou = 0;                                  1
2   struct celula *p = y->prox;                     1
3   while (p != NULL && !achou( {                   1 + 1
4       If (p->chave == X) achou == 1;              1
5       p = p->prox; }                              1
6   if (achou) return p;                            1
7   else return NULL;                               1
                                  Total: T(n) = 8
```

No seu melhor caso, assumimos que o valor procurado seja o primeiro da lista, portanto sua eficiência é de **T(n) = 8** e seu comportamento assintótico é de **O(n) = n.**

O algoritmo promete devolver o valor de p, tal que p->chave == X, porém o algoritmo não está correto, e está devolvendo p = p-> prox.

A correção do algoritmo que sirva para ele atender a ideia proposta, fica a cargo da inserção de uma única condicional **else** dentro do loop, ficando da seguinte forma:

```
1   int achou = 0;                                  1
2   struct celula *p = y->prox;                     1
3   while (p != NULL && !achou( {                   n + 1
4       If (p->chave == X) achou == 1;              n
5       else p = p->prox; }                         n
6   if (achou) return p;                            1
7   else return NULL;                               1
```

2 -

```
ShakeSort(A,n)
1   e <- 1                                          1
2   Para i <- n-1 até e faça                        (n-1)/2
3       Para j <- e até i faça                      n²/4
4           se A[j] > A[j+1]                        n²/4
5               entao troca(A[j], A[j+1])           n²/4
6       Para j <- i até e+1 faça                    n²/4
7           se A[j-1] até e+1 faça                  n²/4
8               entao troca(A[j-1], A[j])           n²/4
9       e <- e+1                                    (n-1)/2
                                    Total: T(n) = 3n²/(2+n)
```

Tomando como base para a análise o pior caso, o algoritmo tem um laço de i até n-1, que servirá como critério de parada, além de mais 2 laços internos, que são de e até i, e outro de i até e+1, portanto, o algoritmo para.

Sua complexidade é de **T(n) = 3n²/(2+n),** portanto seu comportamento assintótico é quadrático, sendo **O(n) = n².**

3 - O algoritmo recebe um número e analisa se ele é primo ou não, tendo como base sendo divisível apenas por 2 números, por 1 e por ele mesmo. Caso ele seja divisível por mais que 2, ele não será primo. O algoritmo possui critério de parada na condição em que ele irá verificar de i até n, portanto, o algoritmo para. Ele possui uma condição em que se for divisível por i, o contador irá incrementar. Ao término do loop, existe outra condicional para verificar se o contador é igual a 2, se for, o número será primo, portanto, o algoritmo está correto.

```
Funcao_Primo(int n) {
1   int i, cont = 0;                                1
2   for(i=1; i<=n; i++) {                           n + 1
3       If (n%i == 0)                               n
4           Cont++;}                                 2
5   If (cont == 2)                                  1
6       return 1;                                   1
7   Else                                             0
8       return 0;                                   0
                                    Total: T(n) = 2n + 6
```

Analisando o algoritmo sendo o número n escolhido sendo primo, ele terá o tempo de **T(n) = 2n + 6** e seu comportamento assintótica será de **O(n) = n.**