	INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA *CAMPUS* CAMPINA GRANDE
Curso: Engenharia de Computação	Disciplina: **Análise e Técnicas de Algoritmos**
Professor: Emanuel Dantas Filho	
Aluno: Pedro Macêdo Luna	

Atividade 03

01) Usando inferência de Laços, realize a prova para o seguinte algoritmo:

```
MAX(A) // Calcula o maior elemento do vetor A
1     i = 1
2     max = A[0]
3     while i < A.size
4           if A[i] > max
5                 max = A[i]
6           i = i + 1
4     return max
```

Invariante:
O máximo elemento de um vetor sempre estará presente ao próprio vetor.

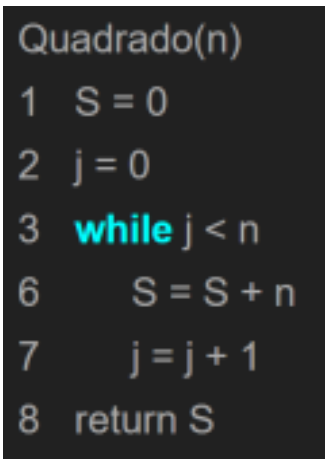
Inicialização:
O maior elemento é inicializado antes do laço com o valor da posição 0 do vetor A.

Manutenção:
Durante o laço, o vetor é percorrido e é verificado se o valor da posição atual é maior que o maior elemento do vetor, se for verdade o valor do maior elemento é substituído pelo da posição atual do vetor.

Terminação:
O maior elemento do vetor A é retornado.

Portanto, o algoritmo está correto!

02) Usando inferência de Laços, realize a prova para o seguinte algoritmo:



Invariante:
O quadrado de um número sempre será um número positivo

Inicialização:
A variável S, que define o quadrado de um número(n) é inicializada antes do laço com 0.

Manutenção:
Caso o número(n) seja maior que a variável j , o laço é executado. Portanto, para o laço ser executado é necessário que o número seja maior que 0. Durante a execução o quadrado cresce por meio da soma do número(n) com o valor da variável S.

Terminação:
É retornado o valor do quadrado caso seja o número declarado seja positivo, caso contrário é retornado o valor 0.

Portanto, o algoritmo está correto!

03) Considere o problema de encontrar o enésimo termo da sequência de Fibonacci. Escreva um pseudocódigo recursivo para esse algoritmo e prove sua corretude por indução fraca.

Código:

```
Fibonacci(n)
   for n in range{0,1}:
     return n
  else:
     return Fibonacci(n-1)  + Fibonacci(n-2)
```

Corretude por indução fraca:

- **Caso Base:**
  Será atribuído o valor 1 para n, devido a condição “if n < 1”. Desse modo, Fibonacci(1) = 1.

- **Passo indutivo:**
  Visto que o valor da propriedade foi correto para o valor de n, precisamos provar que vale para n+1

  Fibonacci(n+1) = Fibonacci((n+1)-1) + Fibonacci((n+1)-2)
  Fibonacci(n+1) = Fibonacci(n) + Fibonacci(n-1)
  Fibonacci(n+1) = Fibonacci(n-1) + Fibonacci(n-2) + Fibonacci(n-1)
  Fibonacci(n) + Fibonacci(n-1) = Fibonacci(n-1) + Fibonacci(n-2) + Fibonacci(n-1)
  Fibonacci(n) = Fibonacci(n-2) + Fibonacci(n-1)

Portanto, a corretude do algoritmo está correta!

04) Analise o seguinte pseudo-código:

```
coisa(l: Array, n: int):
     if n <= 1:
          return l[0]
     if l[0] < l[1]:
          l.remove(1)
     else:
          l.remove(0)
     coisa(l, n-1)
```

O que esse algoritmo faz?
O algoritmo percorre o array l até que o valor de n seja menor ou igual a 1, com o objetivo de descobrir o valor do menor elemento do array.
Para isto é utilizado as seguintes condições durante o laço:

- Caso o valor de n seja igual a 1, deve ser retornado o valor da posição 0 do array.
- Caso o elemento da posição [0] seja menor que o da posição [1], deve ser removido 1 elemento do array.
- Se não, não devemos remover o elemento da posição [0].

Após as condições, é utilizada a recursividade. Passando novamente o array e o valor de (n-1) para n.

Prove sua corretude usando indução fraca.

- **Caso Base:**
  Será atribuído o valor 1 para n, devido a condição “if n <= 1” e o array [4,5,3,1]. Desse modo, temos que:

coisa(**l: [4,5,3,1]**, **n = 1**)
n <= 1, portanto é retornado o primeiro elemento do array, ou seja, o valor 4.

- **Passo indutivo:**
  Visto que o valor da propriedade foi correto para o valor de n, precisamos provar que vale para n+1. Supondo que n >= 1 e o array seja **l: [4,5,3,1]**, temos que:

```
coisa(l: [4,5,3,1], (n+1)):
    if  (n+1) <= 1:
        return l[0]
    if l[0] < l[1]:
        l.remove(1)
    else:
        l.remove(0)
    coisa(l, (n+1) -1)
```

Desse modo, temos que

```
coisa(l: [4,5,3,1], n):
    if  (n+1) <= 1:
        return l[0]
    if l[0] < l[1]:
        l.remove(1)
    else:
        l.remove(0)
    coisa(l, (n))
```