Realizado por: ***Diego Alexander Cardenas Beltrán***

Control de lectura #1

Ejercicios

1. Ordenar las siguientes funciones de menor a mayor orden:

1. n	5. 2^n	9. $n \log n$	13. $\ln n$
2. $n - n^3 + 7n^5$	6. $\log n$	10. $\sqrt{n}$	14. e^n
3. $n^2 + \log n$	7. n^2	11. 2^{n-1}	15. $\log \log n$
4. n^3	8. $(\log n)^2$	12. $n!$	16. $n^{1+\varepsilon}, 0 < \varepsilon < 1$

1) log(log(n))
2) log(n)
3) (log(n))^2
4) ln(n)
5) √n
6) n
7) n log(n)
8) n^2
9) n^2 + log(n)
10) n^3
11) 2^(n − 1)
12) 2^n
13) n!
14) e^n
15) n − n^3 + 7n^5

2. Para las siguientes funciones, determinar el resultado como una función de *n* y representar el peor caso de ejecución con notación Big Oh:

```
function mystery(n)
  r := 0
  for i := 1 to n - 1 do
    for j := i + 1 to n do
      for k := 1 to j do
        r := r + 1
  return(r)
```

```
function pesky(n)
  r := 0
  for i := 1 to n do
    for j := 1 to i do
      for k := j to i + j do
        r := r + 1
  return(r)
```

```
function prestiferous(n)
  r := 0
  for i := 1 to n do
    for j := 1 to i do
      for k := j to i + j do
        for l := 1 to i + j - k do
          r := r + 1
  return(r)
```

★ $\sum_{x=1}^{n} x = \frac{1}{2} n(n+1)$ $\sum_{j=i+1}^{n} j = \sum_{j=1}^{n} j - \sum_{j=1}^{i} j$ $\sum_{x=1}^{n} x^2 = \frac{1}{6} n(n+1)(2n+1)$

1)

```
def mystery(n):                          #Cost          Time(WC)
    r = 0                                #C1                1
    for i in range(1,n-1):               #c2               (n)
        for j in range(i+1, n):          #c3             (n-1)(n-1)
            for k in range(1,j):         #c4             (n-1)(n-2)
                r += 1                   #c5             (n-1)(n-2)
    return r                             #c6                1
```

T(n) = c1(1)+c2(n)+c3((n-1)(n-1))+c4((n-1)(n-2))+c5((n-1)(n-2))+c6(1)
Ecuacion = an^2 + bn +c -> Funcion Cuadratica de n
O(n^2)

2)

```
def pesky(n):                            #Cost          Time(WC)
    r = 0                                #C1                1
    for i in range(1,n):                 #c2              (n+1)
        for j in range(1, i):            #c3             (n)(n+1)
            for k in range(j, i+j):      #c4             (n)(n)(n+1)
                r += 1                   #c5              (n)(n)(n)
    return r                             #c6                1
```

T(n)=c1(1)+c2((n+1))+c3((n)(n+1))+c4((n)(n)(n+1))+c5((n)(n)(n))+c6(1)
Ecuacion = an^2 + bn +c -> Funcion Cuadratica de n
O(n^3)

3)

```
def prestiferous(n):                     #Cost          Time(WC)
    r = 0                                #C1                1
    for i in range(1,n):                 #c2              (n+1)
        for j in range(1, i):            #c3             (n)(n+1)
            for k in range(j,i+j):       #c4             (n)(n)(n+1)
                for l in range(1,i+j-k): #c5           (n)(n)(n)(n+1)
                    r += 1               #c6           (n)(n)(n)(n)
    return r                             #c7                1
```

T(n)=c1(1)+c2((n+1))+c3((n)(n+1))+c4((n)(n)(n+1))+c5((n)(n)(n)(n+1))+c6((n)(n)(n)(n))+c7(1)
O(n^4)

3. Implementar el algoritmo de *insertion sort* para ordenar en orden descendente en vez de ascendente.

```python
def insertionSortElement(new_el, sequence):
    index = 0
    while index < len(sequence) and new_el < sequence[index]: #Se cambia el simbolo del condicional entre new_el y sequence[index]
        index+=1
    return sequence[:index] + [new_el] + sequence[index:]

def insertionSort(sequence):
    for index in range(1,len(sequence)):
        elem = sequence[index]
        sequence = insertionSortElement(elem, sequence[:index]) + sequence[index+1:]
    return sequence
```

Se cambia el símbolo de “>” por el “<” debido a que vamos a empezar desde el mas grande al mas pequeño