**PRACTICO NRO 1. INTRODUCCIÓN A LA ESTRUCTURA DE DATOS**

1.Buscar otra definición de algoritmo en la web.

Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas, además de que es finito.

2. Escribir las instrucciones equivales del lenguaje C++ en Python de: **if, while, do\_until, for, switch**. Haga un pequeño ejemplo da cada uno de ellos.

3. Determine las frecuencias de conteo de lo siguiente, escrito en seudocódigo:

1. for i=1 to n **n+1** 1. i=1 **n**

2. for j=1 to i **i+1** 2. While i<=n do **n+1**

3. for k=1 to j  **j+1** 3. x = x +1 **n**

4. x = x + 1 **n\*j\*m** 4. i = i +1 **n**

5. end **n** 5. End **n**

6. end **n**

7. end **n**

4. La regla de HORNER es un medio para evaluar polinomios:

*A*(*x*) = *anxn* + *an-1xn-1 + ... +a*1 *x + a*0 , en el punto *x*0

Usando el mínimo número de multiplicaciones, la regla es:

*A*(*x*) = (... ((*an*.*x*0 + *an*-1)*x*0 + ... + *a*1)*x*0 + *a*0

Escribir el programa en **Python**

5. Dado n variables booleanas ***x*1,..., *xn***, deseamos imprimir todas las combinaciones posibles de los valores lógicos que son tomadas. Ej. Si **n=2**, hay cuatro posibilidades: **V,V ; V,F; F,V; y F,F** . Escriba un programa en Python para efectuar esto y contar las frecuencias.

6. Comparar la función **n2 and 2n/4** para varios valores de n. Determinar cuando el segundo se hace más grande que el primero.

7. Escribir un p rograma en Python para imprimir valores de x, y, z en orden no decreciente. ¿Cuál es el tiempo computado para su método?

8. Escriba un subprograma en Python en el cual se busca x en un arreglo **A(1:n).** si **x** esta en el arreglo **j** es asignado con la posición en el arreglo, de otra forma j debe estar en cero.

9. Una de las habilidades de Python es la de manipular caracteres y strings, entonces implementar los siguientes procedimientos:

(i) **z = CONCAT(x, y)** concatena **y** luego de **x**, asignando el resultado a **z**

(ii) **z =SUBSTR(x, i, j)** copia a **z** desde la i-esima hasta el j-esimo carácter de **x**. Para j=0, i>j, etc. el string **x** no es cambiado.

(iii) **z=INDEX(x,y)** el cuál busca en **x** la primera ocurrencia de **y**, asignando en **z** la posición inicial en **x, y** cero si no hay la ocurrencia.

Implementar estos procedimientos usando arreglos en Python.

10. Escribir un procedimiento en Python, el cual dado un argumento **STRING**, cuyo valor es un string de longitud n, copie el **STRING** en la variable **FILE**, talque cada secuencia de blancos es reducida a un solo blanco. El último carácter del **STRING** no es blanco.

11. Diseñar un programa en Python que cuente el número de ocurrencias de cada carácter en el string **STRING** de longitud n. Represente su respuesta en un arreglo **ANS**(1:k, 1:2) donde **ANS** (i,1) es el i-esimo carácter y **ANS**(i,2) es el número de veces que ocurre en **STRING**.

12. Probar por inducción.

a)

b)

c)

12. Desarrollar en Python de forma recursiva la serie de **Fibonacci = {0, 1, 1, 2, 3, 5, 8, 13, }** cada termina es la suma de los dos anteriores, comenzando con los dos primero términos de {0, 1} la definición recursiva es:

**Fib(n) = Fib (n - 1) + Fib (n - 2)**

***Caso base:* Fib (0)=0; Fib (1)=1**

***Caso recursivo*: Fib (i) = Fib (i -1) + Fib(i -2), para i>1**

13. Desarrollar en Python de forma recursiva el MÁXIMO COMÚN DIVISOR de dos valores enteros A y B mediante el algoritmo de Euclides. Por Ejemplo:

**MDC**(15,18) es 3.

**MCD**(13, 8) es 1.

**MCM**(12, 24) es 12

14. Desarrollar de forma recursiva las operaciones aritméticas para valores enteros A y B, de:

A+B = **SUMA**(A, B) Ej. SUMA(7, 8) es 15

A\*B = **MULTIPLICA**(A, B) Ej. MULTIPLICA(3, 7) ES 21

A //B = **DIVIDE**( A, B) Ej. PARTE(23, 5) es 4 #división entera

A\*\*B =**POTENCIA**(A, B) Ej. POTENCIA(3, 4) es 81

Por ejemplo la definición recursiva de :

B SI B=0

MULTIPLICA(A, B) =

A + MULTIPLICA(A, B-1)