1. (10) Las listas enlazadas son por definición estructuras de datos homogéneas (es decir, todos sus nodos almacenan elementos del mismo tipo y estructura). ¿Eso es totalmente cierto?, es decir, ¿Sería posible tener una lista que almacene elementos diferentes en cada uno de sus nodos (tanto en tipo como en estructura)? Justifique su respuesta

*En una lista enlazada se pueden agregar cualquier tipo de dato sin importar que otros tipos existan en él, esto se debe ya que estas listas están compuestas por nodos, los cuales a su vez están compuestos por la data y por un apuntador o apuntadores (dependiendo del tipo de lista). Lo importante aquí es que podemos agregar cualquier tipo de dato ya que simplemente el apuntador debe guiarnos hacia donde se encuentra el siguiente sin importar que sea lo que siga.*

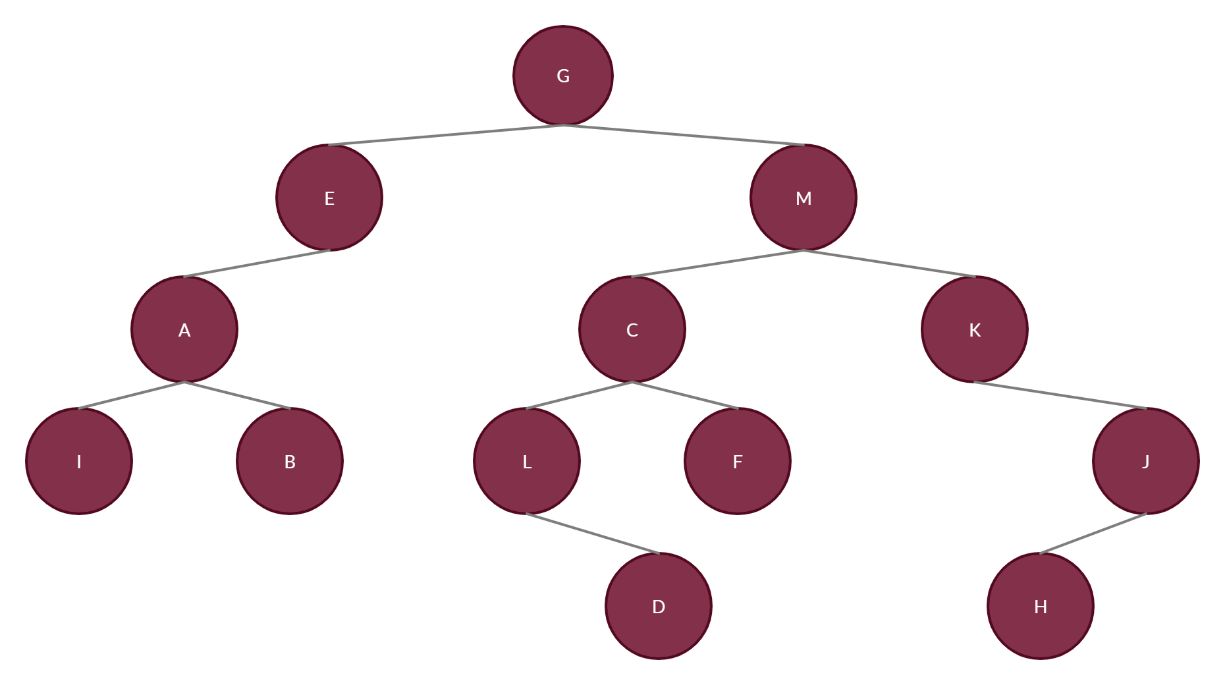
1. (10) Dada una pila, se desea conocer el promedio de los elementos que ella almacena. Como restricción la pila puede ser recorrida una sola vez.

*En código*

1. (15) El recorrido en preorden de un determinado árbol binario es: G E A I B M C L D F K J H y en inorden I A B E G L D C F M K H J .

Resolver:

* 1. Dibujar el árbol binario. (al crear el archivo en GitHub deberá mostrar la figura que quedaría puede ser en Word o Excel)



* 1. Dar el recorrido en postorden.

*I B A E D L F C H J K M G*

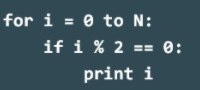
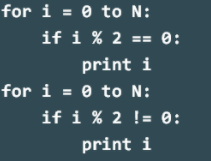
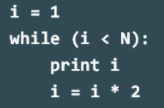
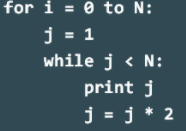
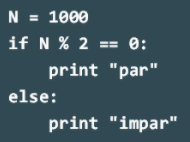
3.3 Diseñar una función para dar el recorrido en postorden dado el recorrido en preorden e inorden y escribir un programa para comprobar el resultado del apartado anterior

1. (10) En McDonald’s se pueden comprar nuggets de pollo en paquetes de 6, 9 o 20 unidades.

Escribir un algoritmo en Python que reciba un parámetro entero y decida si es posible o no comprar ese número de nuggets.

Ejemplos

* es posible comprar 15 nuggets (un paquete de 6 y uno de 9)
* es posible comprar 18 nuggets (3paquetes de 6 o 2 paquetes de 9)
* **Es imposible** compara 14 nuggets

1. (5) Indicar con notación “Big -O”, la complejidad algorítmica (respecto al tiempo) de cada fragmento de código a continuación
   1.  *O(n)*
   2.  *O(n)*
   3.  *O(log n)*
   4.  *O(nlogn)*
   5.  *O(1)*
2. (10) Escribe un programa que calcule la altura máxima de un árbol binario. Si está vacío, la altura se considera 0, y si solo hay la raíz se considera 1.

*En código*

1. (10) En <https://github.com/gsepulv1/ST0245-08/blob/master/taller_Final/estudiantes.py> se encentra un algoritmo que almacena información de estudiantes.
   1. que estructura se utilizó para almacenar los datos de los estudiantes?

*La estructura que se utilizó para almacenar los datos de los estudiantes fue una lista simplemente enlazada*

* 1. que complejidad tiene este algoritmo

*La complejidad de este algoritmo es:*

*10 asignaciones + n iteraciones \* 2 = 10 + n \*2 = 10 + 2n*

* 1. Realice las optimizaciones de código a este algoritmo

*En código*

* 1. Que complejidad tiene después de la mejora realizada en 7.3

*La complejidad nueva seria:*

*3 asignaciones + n iteraciones \* 2 = 3 + 2n*

1. (30) Implemente las modificaciones necesarias al algoritmo anterior y almacene e imprima la información de sus compañeros de curso empleando las presentaciones que se encuentran en el grupo de teams

*En código*