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

Esto es posible aprovechando las similitudes entre las dos estructuras, permitiéndonos hacer este tipo de ‘mezclas’. Sin embargo, mezclar dos tipos de datos en una estructura puede ser difícil dependiendo del tipo. Por ejemplo, mezclar datos categóricos y numéricos en un árbol binario puede ser un problema a la hora de su implementación, ya que ¿cómo compararía un valor numérico con un texto? A pesar de que existen formas de hacerlo, generalmente resulta una tarea difícil.

Por lo tanto, sí es posible mezclar estructuras y tipos de datos.

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.

https://github.com/jfagudelov/ST0245-008/blob/master/punto\_2\_tallerf.py

1. (15)El recorrido en preorden de un determinado árbol binario es: GEAIBMCLDFKJH y en inorden IABEGLDCFMKHJ .

Resolver:

3.1 Dibujar el árbol binario.(al crear el archivo en githun deberá mostrar la figura que quedaría puede ser en Word o excel)

3.2 Dar el recorrido en postorden.

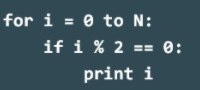
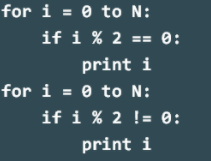
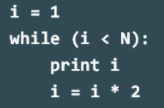
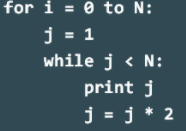
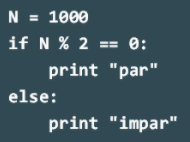
3.3 Diseñar una función para dar el recorrido en postorden dado el recorrido en preorden e inorden y escribir un programa para comprobar el resultado del apartado anterior

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 numero 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(n)
   4.  O(n^2)
   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.

https://github.com/jfagudelov/ST0245-008/blob/master/punto\_6\_tallerf.py

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?

Realmente puede ser ya sea una lista simplemente enlazada o una pila, ya que el nodo definido solo tiene un apuntador al siguiente nodo y esto es aplicado en una lista simplemente enlazada o una pila.

* 1. que complejidad tiene este algoritmo

O(n), por el ciclo que aparece al final.

* 1. Realice las optimizaciones de código a este algoritmo

https://github.com/jfagudelov/ST0245-008/blob/master/punto\_7\_tallerf.py

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

Todavía es O(n), ya que para imprimir la lista sigue requiriendo del ciclo. Sin embargo, es mucho más sencillo añadir nuevo elementos a la pila

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

https://github.com/jfagudelov/ST0245-008/blob/master/punto\_8\_tallerfinal.py