A partir de una estructura **BinaryTree** definida como:

```
class BinaryTree:
      root=None
```

Y una estructura **BinaryTreeNode** definida de la siguiente manera:

```
class BinaryTreeNode:
      key=None
      value=None
      leftnode=None
      rightnode=None
      parent=None
```

EJERCICIO 1

Crear un módulo de nombre **binarytree.py** que **implemente** las siguientes especificaciones de las operaciones elementales para el **TAD árbol binario**.

search(B,element)
Descripción: Busca un elemento en el TAD árbol binario.
Entrada: el árbol binario B en el cual se quiere realizar la búsqueda (BinaryTree) y el valor del elemento (element) a buscar.
Salida: Devuelve la **key** asociada a la primera instancia del elemento. Devuelve **None** si el elemento no se encuentra.

insert(B,element,key)
Descripción: Inserta un elemento con una clave determinada del TAD árbol binario.
Entrada: el árbol B sobre el cual se quiere realizar la inserción (**BinaryTree**), el valor del elemento (**element**) a insertar y la clave (**key**) con la que se lo quiere insertar.
Salida: Si pudo insertar con éxito devuelve la **key** donde se inserta el elemento. En caso contrario devuelve **None.**

delete(B,element)
Descripción: Elimina un elemento del **TAD árbol binario.**
Poscondición: Se debe desvincular el **Node** a eliminar.
Entrada: el árbol binario B sobre el cual se quiere realizar la eliminación (BinaryTree) y el valor del elemento (element) a eliminar.
Salida: Devuelve clave (key) del elemento a eliminar. Devuelve **None** si el elemento a eliminar no se encuentra.

deleteKey(B,key)
Descripción: Elimina una clave del **TAD árbol binario.**
Poscondición: Se debe desvincular el **Node** a eliminar.
Entrada: el árbol binario B sobre el cual se quiere realizar la

eliminación (BinaryTree) y el valor de la clave (key) a eliminar.
Salida: Devuelve clave (key) a eliminar. Devuelve **None** si el elemento a eliminar no se encuentra.

access(B,key)
Descripción: Permite acceder a un elemento del árbol binario con una clave determinada.
Entrada: El árbol binario (BinaryTree) y la **key** del elemento al cual se quiere acceder.
Salida: Devuelve el valor de un elemento con una key del árbol binario, devuelve **None** si no existe elemento con dicha clave.

update(L,element,key)
Descripción: Permite cambiar el valor de un elemento del árbol binario con una clave determinada.
Entrada: El árbol binario (**BinaryTree**) y la **clave (key)** sobre la cual se quiere asignar el valor de **element.**
Salida: Devuelve **None** si no existe elemento para dicha clave. Caso contrario devuelve la clave del nodo donde se hizo el update.

EJERCICIO 2

Implementar las siguientes especificaciones de las operaciones para recorrer un TAD árbol binario. Incluir las implementaciones en el modulo **BinaryTree.py**

traverseInOrder(B)
Descripción: Recorre un árbol binario **en orden**
Entrada: El árbol binario (**BinaryTree**)
Salida: Devuelve **una lista (LinkedList)** con los elementos del árbol en orden. Devuelve None si el árbol está vacío.

traverseInPostOrder(B)
Descripción: Recorre un árbol binario **en post-orden**
Entrada: El árbol binario (**BinaryTree**)
Salida: Devuelve **una lista (LinkedList)** con los elementos del árbol en **post-orden**. Devuelve None si el árbol está vacío.

traverseInPreOrder(B)
Descripción: Recorre un árbol binario **en pre-orden**
Entrada: El árbol binario (**BinaryTree**)
Salida: Devuelve **una lista (LinkedList)** con los elementos del árbol en **pre-orden.** Devuelve None si el árbol está vacío.

traverseBreadFirst(B)
Descripción: Recorre un árbol binario **en modo primero anchura/amplitud**
Entrada: El árbol binario (**BinaryTree**)
Salida: Devuelve **una lista (LinkedList)** con los elementos del árbol ordenados de acuerdo al modo **primero en amplitud**. Devuelve None si el árbol está vacío.

A tener en cuenta:

1. Cada operación básica debe ser implementada como una función.
   Ejemplo:

```
def insert(B,key,element):
      Aca va el código que implementa la operación insert
```

2. Las operaciones deben respetar la especificación propuesta. Es decir sólo incluir los parámetros mencionados en la definición de la operación.
3. Usar lápiz y papel primero.
4. No se puede utilizar otra biblioteca/módulo que los desarrollados en clase:
   a. algo1.py
   b. array.py
   c. linkedlist.py
   d. queue, stack
   e. etc.