Exercise Session n. 10 (5 May 2023)

Algorithms and Data Structures

In-Order Walk in a BST: Non-Recursive Variant n. 1

The following Python code defines the structure and the insertion algorithm for a binary search tree (BST).

```python
class Node:
    def __init__(self, k):
        self.key = k
        self.parent = None
        self.left = None
        self.right = None

def tree_insert (t, k):
    # return the root of the tree we obtain from inserting a new
    # with key k into tree t
    if t == None:
        return Node(k)
    p = t
    while True:
        if k <= p.key:
            if p.left == None:
                p.left = Node(k)
                p.left.parent = p
                return t
            p = p.left
        else:
            if p.right == None:
                p.right = Node(k)
                p.right.parent = p
                return t
            p = p.right
```

Write a non-recursive function `print_in_order(t)` that prints on a single line all the keys in a BST rooted at t in their natural order (sorted).

Examples

```
>>> t = None
>>> for k in [7, 1, 3, 5, 10, 2, 1, 5]:
...     t = bst_insert (t, k)
...
>>> print_in_order (t)
1 1 2 3 5 5 7 10
```

In-Order Walk in a BST: Non-Recursive Variant n. 2

The following Python code defines the structure and the insertion algorithm for a binary search tree (BST).

```
class Node:
    def __init__(self, k):
        self.key = k
        self.left = None
        self.right = None

def tree_insert (t, k):
    # return the root of the tree we obtain from inserting a new
    # with key k into tree t
    if t == None:
        return Node(k)
    p = t
    while True:
        if k <= p.key:
            if p.left == None:
                p.left = Node(k)
                return t
            p = p.left
        else:
            if p.right == None:
                p.right = Node(k)
                return t
            p = p.right
```

Write a non-recursive function `print_in_order(t)` that prints on a single line all the keys in a BST rooted at t in their natural order (sorted).

Examples

```
>>> t = None
>>> for k in [7, 1, 3, 5, 10, 2, 1, 5]:
...     t = bst_insert (t, k)
```

```
...
>>> print_in_order (t)
1 1 2 3 5 5 7 10
```

Hints

Notice that this BST variant does not have `parent` links. That is, a node t does not store a parent link. In this case, you might want to use an auxilary data structure such as a *stack* to keep track of the *path* you are currently on as you walk through the BST.