Hlutapróf 3 - programming assignments.

There are two assignment descriptions on the next two pages.
Implement both of them in the file ***solutions.py*** that you can find in a **ZIP** archive in the assignment description on *Canvas*.
Hand in only the **PY** file ***solutions.py***

1. LRCMap (25%)
2. HashMap (25%)

There are multiple choice questions in a quiz in Canvas.

3. Multiple choice (50%)

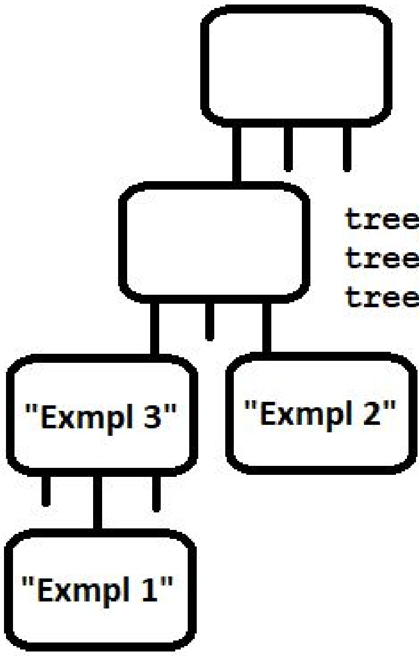
1. Make a tree in the class ***LRCMap*** (**25%**)
The tree node can have *references to other tree nodes* and a single data variable, ***nothing else***. The tree node can ***not*** hold the key value itself. The key must be represented only by the placement of the node in the tree. *There can be data in any node, not only leaves*.
The keys are strings which only have three possible characters, but as many of those as needed. The characters are **'l'** (*left*), **'r'** (*right*) and **'c'** (*center*), so an example is: “llrcrclcclr”.
The tree class has two operations:

- put_data(key, data)
  - Places this ***data*** in the tree corresponding to this ***key***
  - Overwrite/update if data is already there.
- get_data(key)
  - returns ***data*** for that ***key***
  - returns ***None*** if non-existant
  - Returns as soon as it is evident the key is not there
    - Does not initialize any new nodes while searching

The constructor takes a boolean parameter.

- __init__(self, build = False)
  - If ***build*** is set to True, build a tree that can be used for strings of length up to 8 characters, without adding new nodes after the initialization.
  - If ***build*** is set to false, initialize an empty tree, or root only.

Example input and tree:



```
tree_instance.put_data("llc", "Exmpl 1")
tree_instance.put_data("lr", "Exmpl 2")
tree_instance.put_data("ll", "Exmpl 3")
```

2. The class ***HashMap*** with is given with an implemented **___init___** function.
Two of the lines are commented out and students can choose between these two lines. Uncomment the one you choose. ***You can not change*** __init__ ***in any other way***. You can add helper functions, but must use either ***list*** or ***dict*** as buckets, as per the line you choose.

```
def __init__(self):
    self.array_length = 16
```
Choose one of these:
```
  #  self.hash_table = [ [ ] for _ in range(self.array_length) ]
  #  self.hash_table = [ { } for _ in range(self.array_length) ]
    self.item_count = 0
```

Finish these implementations:

- def __setitem__(self, key, data)
    - Adds this data connected to this key
    - *overwrites/updates if already there*
- def __getitem__(self, key)
    - returns data for the key
    - *returns* **None** *if nothing there*
- def __len__(self)
    - returns the number of items currently in the map