CPS305 Lab 4

Topics:

- Huffman Coding

Files:

- Lab4.py
- Incorrectly named files will receive a mark of zero.

Submit Files:

- Submit file through D2L
- Your file should not have any print statements in it.
- You will lose a mark for every print statements.
- There is a getName() function in the Lab4.py file.
- Change the output string to your last and first name.
- Your name should be EXACTLY as written on D2L.
- Incorrectly filed out getName function result in a mark of zero.

Lab Description:

In this lab, you are implementing a solution for Huffman codes. You are given a Python Class template. In this template, there 2 classes: Node, MyHuffman. The goal of this lab is to implement a Huffman class that can encode and decode text.

Requirements:

- First, create a method called getName() that returns your name. If this does not work, you will receive a mark of zero.
- A class called MyHuffman is given. Implement a method called build(x) where x is a dictionary of characters to integers, in which the integer is the frequency of that character. The method should build a Huffman tree based on these given frequencies. Your Huffman tree should assign the left edge to be "1" and "0" for the right edge. The smaller frequency should be on the left edge.
- Implement a method called makeLookupTable (). This method should be called after the completion of building the Huffman tree. This method should use a recursive algorithm to build a dictionary of the characters and their corresponding bitcode. This is useful for encoding. This is not called by the tester so you can modify the parameters as desired.
- Implement a method called encode(x) that takes the string x and returns a bitstring by applying the rules in your Huffman tree. A bit string is a standard string with only the characters '0' and '1'.
- Implement a method called decode(x) that takes the bitstring x and returns a string by applying the rules in your Huffman tree. Use the recursiveTraverseTree() method described below to decode.
- Implement a method called recursiveTraverseTree(). This method should utilize a recursive algorithm to traverse the Huffman tree to find the appropriate corresponding character.

Restrictions:

- The structure of the tree should adhere to the efficiency of Huffman's algorithm. Implementation details are up to you.
- The given Node class is there if you choose to use it.

Testing Your Code:

You can test your code with the tester. The tester is not a be all and end all. You should rely on your own testing before using the tester. Note that the lab should be done on Python 3 and the tester only works on Python 3. The tester is named Lab4Tester.py. Your file must be named correctly and be in the same folder of the tester. Here is an example of executing it on the command line (note that the $ is representative of the beginning of the new line).

```
$ python3 Lab4Teser.py
```