Implementation of

Merkle Tree Proof of Inclusion and Consistency

# Introduction

The Merkle Tree data structure is comprised of a binary tree where leaf values are data nodes, parents are conjoined hashes of their children, and the root node is the entire hash of all elements in the tree. The proof of inclusion and consistency serve a purpose to ensure that data integrity is maintained throughout the lifespan of additions to this data structure

# Environment

For my environment, I am running PyCharm 2022.3.2 Professional Edition on Windows 11 Home version 22H2. I have treelib and hashlib imported to help with the generation of the Merkle tree

# Usage

I have generated 3 files for the grader:

1. buildmtree.py
2. checkconsistency.py
3. checkinclusion.py

## buildmtree.py Functional Description

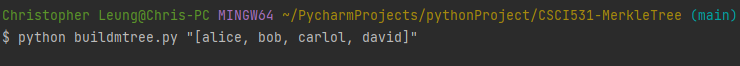
### Usage:

python buildmtree.py "[<data1,data2, ..., datan>]"

### Description:

This python script builds a full binary tree by passing in a list of names from the command line interface. This program outputs a file (merkle.tree) which can be read using a text editor to determine all nodes, hashes, and root nodes that are contained within Merkle tree.

### Screen Captures:



Text

Description automatically generated

### Note to grader:

The Merkle Tree generation will ensure that the tree generated is full. It will copy the last leaf node certain amount of times to obtain a power of 2. Please see piazza post below for verification of concatenation of non-full leaf nodes to complete the binary tree.

Piazza post <https://piazza.com/class/lcgjdtutgvz7o9/post/149>

## checkinclusion.py Functional Description

### Usage:

python checkinclusion.py <name>

### Description:

This python script builds parses merkle.tree and finds the minimum number of nodes required to verify the root hash.

### Screen Captures:

Graphical user interface, text

Description automatically generated

## checkconsistency.py Functional Description

### Usage:

python checkinclusion.py "[<data1,data2, ..., datan>]" "[<data1,data2, ..., datan>]"

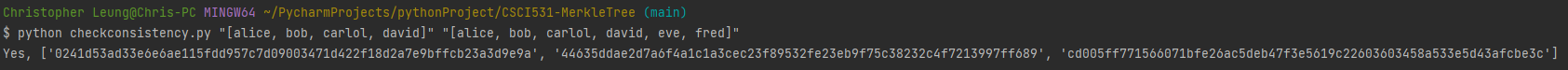
**Note**: Please use quotations (“ “) and brackets ([]) to denote Tree 1 and Tree2 separately

### Description:

This python script takes in 2 lists and determines to see if the first one is a subset of the second

### Screen Captures:

Test 1 “[alice, bob, carlol, david]” “ [alice, bob, carlol, david, eve, fred]”



Text

Description automatically generated

Text

Description automatically generated

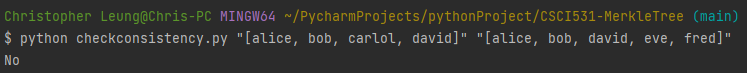
Text

Description automatically generated

Text

Description automatically generated

Test 2 "[alice, bob, carlol, david]" "[alice, bob, david, eve, fred]":



Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

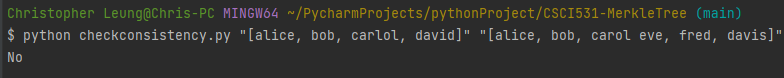
Text

Description automatically generated

Text

Description automatically generated

Test 3 “[alice, bob, carlol, david]” “[alice, bob, carol eve, fred, davis]”



Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated