**Data Model Tree**

**Background:**

The configuration data inside of Drivenets router has a pre-defined tree-like structure (a YANG schema). It is the team’s responsibility to model and distribute this data in the system.     
In this question we will design and implement a data structure to model the router’s data and its changes over time.

**Terminology:**

Each **node** in the tree has a **name,**  **path**  and can hold a **value**.

The  **path**  leads from the tree’s top to the node itself.  Each node along the path is separated from the next node with a slash ‘/’ (just like a file-system path).   
**Example**: ‘dn-top**/**Protocols**/**Protocols2/leaf-id’

Diagram

Description automatically generated

**Assumptions:**

All throughout the question you may assume:

* You may use any standard data structures and functions
* Any function for splitting a string is available.
* The tree’s schema is not known in advance. i.e, the tree is only constructed via the SetNodeValue api below
* Please discuss any other assumptions you’re making

Question 1:

* Please design the structure of the tree data structure

|  |
| --- |
| def SetNodeValue(path: str, value: int):      """      Sets a node’s value.      Any non-existing nodes in the path will be created if necessary (like mkdir –p <path>)      :param path: Path of the node in the tree. Names of nodes saparated by '/'      :param value: int value to set to the node      :return: TBD      :return: Running and space complexity?      """      pass    def GetNodeValue(path: str) -> int:      """      Gets a nodes value from the tree      :param path: Path of the node in the tree. Names of nodes saparated by '/'      :return: TBD      """      pass    def print\_tree(path: str):      """      Print the sub-tree rooted at and including the node located at the path      If a node has a value, it will be printed as "{name}:{value}"      If a node does not have a value, it will be printed as "{name}"      :param path: Path of the node in the tree. Names of nodes saparated by '/'      :return: TBD      """      pass    ##################################################################### |
| SetNodeValue('dn-top/Protocols/Protocol1/leaf\_priority', 1)  SetNodeValue('dn-top/Protocols/Protocol1/leaf\_id', 2)  SetNodeValue('dn-top/Protocols/Protocol2/leaf\_id', 3)  SetNodeValue('dn-top/Interfaces/Interface2/leaf\_is\_first', 4)  SetNodeValue('dn-top/Interfaces/Interface2/leaf\_is\_first', 5)  GetNodeValue('dn-top/Interfaces/Interface2/leaf\_is\_first') # Prints 5  # Last ‘set’ wins  print\_tree()  dn-top    Protocols      Protocol1        leaf\_priority = 1        leaf\_id = 2      Protocol2        leaf\_id = 3    Interfaces      Interface2        leaf\_is\_first = 5 |

* Please design and implement the following API

Question 2:

We would now want to add simple versioning capabilities to the tree. This is similar to git commit and git diff

Expend the API by adding:

def save():

    """

    Save/Remember the tree’s current state.

    """

    pass

def printDiff():

    """

    Print the difference (i.e. changed nodes and their values) between the current tree’s state and the last saved state.

    """

    pass

Question 3:

We would now want to add an option to fall back to the last saved configuration. This is similar to git stash

Expend the API by adding:

def rollback():

    """

    Revert the tree’s current state to the last saved state.

    """

    pass

Question 4:

Assume that each save is remembered as a distinct version of the tree (like in source-control).   
  Expend the API by adding:

def rollback(n: int):

    """

        Revert the tree’s current state ‘n’ saved-state backwards.

    """

    pass