CS2123 Doc-In-A-Box using Binary Tree (60 pts)

In this program you will store a diagnosis tree in a binary tree which will contain nodes that are either:

* Question - a question has an ID which refers to a question text. The question's answer will be either Y or N. If N, follow pLeft. If Y, follow pRight. The nodes to the left and/or right could be either Question or Diagnosis nodes.
* Diagnosis - a diagnosis has an ID which refers to a diagnosis text. The pLeft and pRight pointers should be NULL for a diagnosis since it is a leaf.

The nodes will **not** be in ascending order; instead, they are in an order that maintains the diagnosis tree. See the diagram for an example.

Input: Command Data File (read until EOF) containing one command per data line:

ROOT qId Defines a question node as the root of the tree. You may assume that there will be only one ROOT command in the file.

NODE type id parentQId yn Defines a node (other than root) in the tree. It includes:

type - Q (question) or D (diagnosis)

id - ID for this question or diagnosis

parentQid - the ID of the question which is its parent

yn - Y (this is the Yes branch) or N (this is the No branch)

Confirm that **id doesn't** already exist in the tree (use searchT) and that the **parentQid does** already exist. If either of those is a problem, display a warning and ignore this command.

TEXT type id text Provides the display text for the specified question or diagnosis.

PRINT Prints the tree in a pretty print style. This should print the ID and corresponding display text.

HELP answers The answers are a string of Y and/or N answers. It is not a problem for there to be more answers than questions for a particular path. (Do not treat that as an error/warning.) As the diagnosis tree is traversed, display the question ID, its display text, and the answer (Y or N). The result of HELP should cause a diagnosis ID and its text to display. If there isn't a diagnosis, return NULL. See the sample HELP output.

DELETE id This causes the specified node to be deleted from the tree (**its parent should no longer reference it**). It and its descendants must be freed. You may assume that we will NOT delete the root.

\* Defines a comment in the data. It is only used for helping explain the data.

Notes:

1. The binary tree represents the diagnosis tree. It is **NOT** an ascending ordered binary tree.

2. Use the provided data to build a diagnosis tree for Doc-In-A-Box. It must be a binary tree. **Recursion must** be used when inserting data into the tree and traversing the tree.

3. If the NODE command references a branch of its parent which already has been populated, show a warning that an answer has already been provided.

4. Write **recursive** functions:

searchT search for an ID in the tree and return a pointer to the NODE or NULL if not found

prettyPrintT print the contents of the tree in a pretty print style

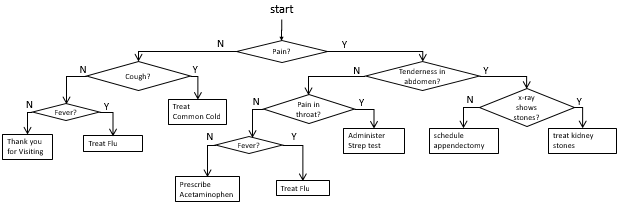
help returns the diagnosis based on the answers to the questions. If it doesn’t have one, it returns NULL.

freeT frees all nodes in this subtree. It is assumed that the subtree has already been disassociated from the tree.

The insertion function must also be recursive, but its prototype is not provided. There are additional functions that you may need which might not be recursive. Modularity matters!

5. The most difficult capability is providing DELETE id.

6. A data command-oriented driver has been provided, but it is skeletal (incomplete). You need to complete it.



Sample Help output:

HELP YNYNYNYN

Q1: Pain? Y

Q5: Tenderness in abdomen? N

Q6: Pain in Throat? Y

D7: Administer Strep Test