Program 5 - Price Menu Application

Create a program to manage a Price Menu for the Klunker Car Company using a binary tree to represent the information.

Example Price Menu Information:

MODEL (OPTION) BASE $17000 ENGINE (OPTION) 1.8-liter 4 Cyl Automatic $0

1.8-liter 4 Cyl 6spd Manual $100

COLOR (OPTION) WHITE $0

BLUE $50

AUDIO (OPTION) AUDIO STD $0

8-TRACK $-100

PLUS $19000 ENGINE (OPTION) 1.8-liter 4 Cyl Automatic $-100

1.8-liter 4 Cyl 6spd Manual $0

COLOR (OPTION) RACING WHITE $0

DEEP BLUE $0

METALLIC COAL $50

AUDIO (OPTION) AUDIO STD $0

SURROUND $200

OY $26000 ENGINE (OPTION) Turbo $0

Turbo Elite $2500

COLOR (OPTION) BLACK PEARL $-50

SEA GREEN $0

EMPRESS $100

AUDIO (OPTION) SURROUND STD $0

SURROUND PREMIUM $250

WARRANTY (OPTION)

40k/3yr $500

55k/4yr $700

70k/5yr $900

NODE TYPES

OPTION

An **option node** has children which are the option values. For example, MODEL is an option node. Its first child (i.e., its first option value) is BASE. Option nodes may also have a pointer to other options (e.g., MODEL has a sibling pointer to WARRANTY.

Pointers in Option nodes:

pChild - points to an option value node (e.g., MODEL points to BASE). This would be the first option value in a list of option values.

pSibling - points to other option nodes (e.g., MODEL points to WARRANTY, ENGINE points to COLOR)

OPTION VALUE

An **option value node** represents one of the values for an option. For MODEL, there are three option values: BASE, PLUS, OY. Option Value nodes have a sibling pointer which point to the next Option Value. Additionally, Option Value nodes have a child pointer which points to subordinate features associated with this option value.

Pointers in Option nodes:

pSibling - points to next option value (e.g., BASE points to PLUS, PLUS points to OY)

pChild - points to a subordinate feature associated with this node (e.g., BASE points to ENGINE)

Command input file

DEFINE OPTION szId szSubordinateToId szTitle

where szId is the ID of this new node,

szSubordinateToId is the ID of the node that is its parent (via a pChild pointer),

szTitle is the title for this option (e.g., "Model").

Assuming the node with szSubordinateToId exists, this inserts a new node in the tree. If a child already exists, follow that child's sibling chain until you can insert it at a pointer that would be NULL.

Note: if we are inserting at the root, the szSubordinateToId will be "ROOT".

Print warnings (don't terminate) if the szId already exists or if the szSubordinateToId doesn't exist.

DEFINE VALUE szId szOptionId cCostInd dCost szTitle

where szId the ID of this new node,

szOptionId the ID of the option which is its parent (via a pChild pointer),

cCostInd character (not a string) representing whether it has a cost,

dCost the cost which may be positive or negative,

szTitle the title for this option value (e.g., "Racing White").

Assuming the **option node** with szOptionId exists, this inserts a new node in the tree. If a child already exists for that Option, follow that child's sibling chain until you can insert it at a pointer that would be NULL.

Print warnings (don't terminate) if the szId already exists or if the szOptionId doesn't exist. If the node containing the szOptionId isn't an OPTION, print a warning.

PRINT ALL

prints the entire tree in a pretty print style (see sample output).

PRINT ONE szId

prints one item. Print a warning if the szId doesn't already exist.

QUOTE BEGIN

starts a quote selection. Note that this shouldn't call determineQuote.

QUOTE OPTION iLevel szOptionId iSelection

…

QUOTE END

This describes a set of values used to get a price quote. It should invoke determineQuote passing

the information from the QUOTE OPTION commands. See the discussion about determineQuote and the sample output.

Example:

QUOTE BEGIN

QUOTE OPTION 0 MODEL 1 // selected BASE

QUOTE OPTION 1 ENGINE\_BASE 1 // selected 1.8-liter 4 Cyl Automatic

QUOTE OPTION 1 COLOR\_BASE 2 // selected BLUE

QUOTE OPTION 1 AUDIO\_BASE 2 // selected 8-TRK

QUOTE OPTION 0 WARRANTY 3 // selected 70k/5yr

QUOTE END

It would print a quote:

MODEL BASE 17000

ENGINE 1.8-liter 4 Cyl Automatic 0

COLOR Blue 50

AUDIO 8-Track -100

WARRANTY 70k/5yr 900

Total 17850

DELETE szId

This causes the specified node to be deleted from the tree (as a result, **its parent should no longer reference it**). It and its descendants must be freed. Do not delete its immediate siblings. For example, deleting BASE should remove it from its parent's value chain, but should not cause PLUS and OY to be deleted.

Print a warning if the szId doesn't already exist.

\* a comment in the data. It is only used to explain the data.

Notes:

1. A binary tree represents the Price Menu Tree. It is **NOT** an ascending ordered binary tree. The ordering is based on the relationship between nodes.

2. Initially, use the provided data to build a Price Menu Tree. You will need to create additional data to fully test your code.

3. Recursive functions:

NodeT \*findId(NodeT \*p, char szId[])

search for a specified szID in the tree and return a pointer to the node containing that ID. If not found, return NULL.

NodeT \*findParent(NodeT \*pParent, NodeT \*p, NodeT \*pkid)

search for a pointer to a kid in the tree and return a pointer to the logical parent (might not be the physical parent). pKid is the pointer we are trying to find. If not found or if the kid is at the root level, return NULL. The logical parent of the kid which points to OY is the MODEL. pParent is initially NULL and changes as we follow a child pointer. (pParent should not change when you follow a sibling pointer.) p is the current position in the traversal.

void freeSubTree(NodeT \*p)

frees the nodes in a subtree of the tree. This assumes that other pointers no longer point to it.

4. There are additional functions that you will need which *might* not be recursive (your choice). You may find it useful to add recursive *helper* functions. It is possible that these might use functions from note #3.

void insertPriceMenu(Tree tree, Element element, char szParentId[])

inserts one menu item into the tree. This is used from the DEFINE command.

void deleteItem(Tree tree, char szId[])

deletes one item (and its pChild children )from the tree. It should not delete its siblings. The deleted nodes must be freed. This is used from the DELETE command.

void printPriceMenu(Tree tree)

prints the contents of the tree in a readable style (see sample output)

void printOne(Tree tree, char szId[])

prints one item from the tree. This is used from the PRINT ONE command.

void freeTree(Tree tree)

frees all the nodes in the tree. This probably uses freeSubTree.

QuoteResult determineQuote(Tree tree, QuoteSelection quoteSelection)

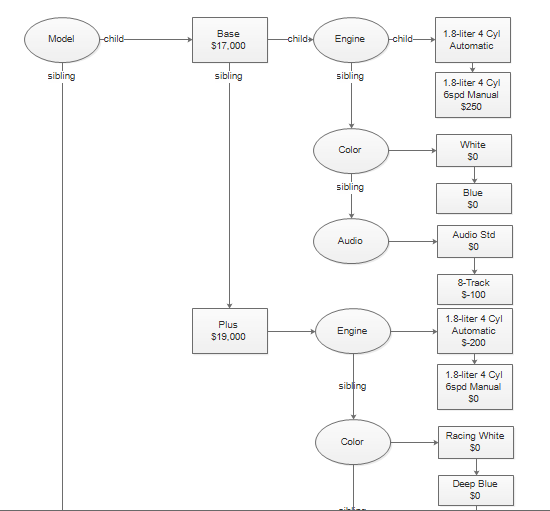
after the QUOTE END command is received, this is called. determineQuote:

* Returns a QuoteResult which includes a total cost
* Prints the quote details including the title and cost for each option value
* Handles error cases and understands a partial quote

Note that the driver must examine the returnCode and print whether this was a total cost, partial cost, option error, or option selection error. For total cost and partial cost, it should also print those totals.

5. A simple main driver, cs2123Driver.c, is provided for everyone. Additionally, cs2123p5DriverHelper.o is provided to indiviuals. Groups must create the driver themselves. Individuals should see the document titled "Program 5 for Individuals".

In the partial diagram below, ovals represent Option nodes and rectangles represent Option Value nodes.



See the sample output file.

**What to turn in?**

Hwk 5.1: printed sheet (one per group; "selfs" must also turn in the sheet). It must be given to Larry at the beginning of class on the due date. (Do not submit via BlackBoard)

Pgm 5.2 and pgm 5.3: via upload in BlackBoard

* team leader (for groups) must turn in the following as a single zip file:
  + all .c files
  + .h file(s)
  + makefile
  + output
  + instruction to TA sheet (explaining who is on the team (last name, first name) and how to compile/execute your code)
* Other participants in a group:
  + instruction sheet explaining who is on the team (last name, first name) and who is submitting the code
* "selfs" (teams of 1):
  + cs2123p5.c
  + cs2123p5.h (if it changed)

Hwk 5.4: via upload in Blackboard (everyone)

* Group Evaluation form as a PDF
* Peer Evaluation form as a PDF