Splay Trees

Assignment:

For this project, you will implement a simple database system with multiple key indices. Specifically, the records in the database will contain three pieces of information about cities: Their population, their economic "net worth" and their name. The records will be organized for search by a collection of three search tree structures. The trees you will implement are a variant of the Binary Search Tree called the splay tree. The splay tree is essentially a modification to the standard BST insert/delete/search routines that guarantee good performance over a series of operations. Read Section 13.2 of the textbook regarding balanced trees for more information (read the section on AVL trees as well as the one on splay trees).

Implementation:

Records in the database each contain three fields. The *population* and *economy* fields are integer values. The *name* field is a variable length string beginning with a letter and containing any number of letters (upper and lower case) and digits. Names are case sensitive. You may (and should) compare names for equality using the standard `strcmp` function or something equivalent.

Your database will contain three separate splay trees. Each splay tree node will contain a pointer to the appropriate data record, and the various splay trees are distinguished by their comparator functions. It is recommended, but not required, that you derive your splay tree implementation from the Binary Search Tree and binary tree node classes provided as part of the class textbook source code. It is **required** that your program include the source code file `dictionary.h` without any modifications, and that your splay tree class inherit from the Dictionary class.

All splay tree operations must be implemented recursively. That is, any function that traverses or otherwise moves through the tree must be recursive. There will be a minor point deduction if your node class uses parent pointers. Note that on an unsuccessful search, the last node encountered will be the object of the splay operation. For example, in Figure 13.10, the same splay operations would take place if the value 89.5 were searched for in the tree of part (a).

Input and Output:

The input to this program should be **read from standard input** and the output should be **directed to standard output**. The name of the program should be "splay".

The input for this project will consist of a series of commands (some with associated parameters, separated by spaces), one command for each line. Commands are free format in that an arbitrary number of additional spaces may be interspersed between parameters. You do not need to check for syntax errors in the command lines, however you do need to check for logical errors (such as deleting a value that is not in the tree).

Each input command should result in meaningful feedback in terms of an output message. Each command's parameters should be printed, and in addition, some indication of success or failure of the operation should be reported. Some of the command specifications below indicate particular additional information that is to be output. Some of the commands refer to a specific data field, by number. Field "1" is population, field "2" is net worth, and field "3" is name.

Commands and their syntax are as follows.

insert *population economy name*

Insert a record into the database. You will create a new record object, and insert it into each of the three splay trees.

No record is permitted to duplicate the value of the same field in any other record. For example, its is permitted for the database to contain the two records "1000 12000 Atown" and "12000 1000 Btown" but it is **not** permitted for the database to contain the two records "1000 12000 Atown" and "1000 15000 Btown". If an attempt is made to insert a record with a duplicate field value, then the insert attempt should be rejected. Note that this might require you to reverse the insertion already done on some tree(s) for that record.

find *field value*

Print some record with value *value* in field *field*.

remove *field value*

Remove some record with value *value* in field *field* from the database, if it exists. To do so, you will need to remove the record from all three splay trees.

list *field*

List out the records in the database, sorted by *field*. The listing should be processed by an inorder traversal of the appropriate tree, with each record printed on a separate line. If the record appears in a node at depth i in the tree, it will be printed starting in character column $4i$ on the page (counting the first column as 0).

makenull

Reinitialize the database to be empty.