

CS 2213, Spring 2009
Assignment 7: Phone Book 1I
Due: 5pm, Monday, April 13, 2009

Goal: The purpose of this assignment is to learn to use AA Trees to implement the Dictionary ADT.

Objectives: be able to write code to implement insertion into an AA Tree; be able to write code to implement lookup in an AA Tree; be able to write code using the `strcmp` function.

Phone Book Application

Write a program `phonebook2` that will read in a file containing pairs of lines names and phone numbers, place those into an AA Tree. and then read in a list of names (one per line) from the standard input. For each name read from the standard input, the program should determine if there is an exact match in the AA Tree. If there is a match it should print out the name, followed by the phone number. If there is not a match, it should print out the name followed by the string `"No_match_found."`.

Input/Output Specification

The program should work with input and produce output exactly as described here. (Automated tools that do a character by character comparison may be used for grading.) For inputs that do not comply with this specification, your program may print an error message and exit; there should not be any inputs which would cause your program to have undefined behavior.

phonebook2 command line. The program executable should expect as a command-line argument the name of a file containing a list of phone numbers. Example:

```
./phonebook2_phonelist.txt
```

data file. The file specified on the command-line (e.g., `phonelist.txt`) will contain lines of up to 80 non-null characters. (If there are more than 80-characters—not counting `'\n'`—on a line, the additional characters should be silently discarded, producing the same result as if all of the lines were truncated to 80 characters.)

Each pair of lines will represent one 'phonebook entry'. The text of odd lines will be considered to be a person's name. The text of even lines will be considered to be the phone number of the person named on the previous line. The program should treat these items as uninterpreted strings of up to 80 non-null characters. (In particular the program should **not** try to parse the name to determine the last name or parse the the phone number to encode the individual digits.) The entries will not necessarily be sorted in any way.

An example data file is:

```
John_Smith
+49_1234_5-67890
Jeff
458-5667
Abe_Lincoln
1-888-WIT-HOUS
```

Standard Input. The program should interpret each line on the standard input as a name that it should attempt to find in the list. (If there are more than 80-characters + '\n' on a line, the additional characters should be discarded.) The program should stop when it sees a blank input line or EOF.

An example of some input to the program would be:

```
Jeff
Dr._von_Ronne
Supercalifragilisticexpialidocious
John_Smith
Abe
Jeffery

Bubba
```

For testing, you may wish to redirect the input from files containing lists of names such as that found in this example.

Standard Output. For each of the lines on the standard input, the program should attempt to find a phonebook entry who's name is an exact match of the name in the standard input line. If there is a match, the program should output the name, a colon and space, and then the phone number on a single line. If there is no match, it should output the name, a colon, and the text, "No_match_found."

The output for the example inputs listed above would be:

```
Jeff:_458-5667
Dr._von_Ronne:_No_match_found.
Supercalifragilisticexpialidocious:_No_match_found.
John_Smith:_+49_1234_5-67890
Abe:_No_match_found.
Jeffery:_No_match_found.
```

Implementation

The specification above is identical to that of assignment 5, but in this assignment you should store the names and numbers of the phonebook entries into an AA Tree data using the name string as a key and the number string as the value. You should use the ordering provided `strcmp` to compare keys in your AA Tree implementation. Insertion and lookup should be implemented

using the standard algorithms that operate in $O(\log n)$ time and correctly maintain the AA Tree invariants.

Provide a Makefile that produces an executable `phonebook2` with the proper dependencies on your source files.

Test your program thoroughly using `valgrind` and a variety of valid and invalid inputs.

Extra Credit: Deleting, Control Language

Implement a function to delete the node for a particular key in the AA Tree. This function should operate in $O(\log n)$ time and maintain all of the AA Tree invariants.

Create an alternative main source file and makefile rules to produce an executable `phonebook2-extra-credit`.

The input for `phonebook2-extra-credit` should be from the standard input and consist of a sequence of `insert`, `lookup`, and `delete` commands formatted as in the following example:

Input:

```
add John Smith : +49 1234 5 -67890
add Jeff : 458-5667
lookup John Smith
delete Jeff
lookup Jeff
add Abe Lincoln : 1-888-WIT-HOUS
lookup John Smith
lookup Abe Lincoln
```

Output:

```
John Smith : +49 1234 5 -67890
Jeff : No match found.
John Smith : +49 1234 5 -67890
Abe Lincoln : 1-888-WIT-HOUS
```

Please note what you did to solve this problem in the EXTRA CREDIT section of the README. Please also describe what you did to validate the correct operation of your delete function.

Submission

Please create a README file divided into three sections: NOTES TO GRADER, EXTRA CREDIT (optional), PROGRAM VALIDATION, and CERTIFICATION.

1. Under NOTES TO GRADER, add any notes, you wish the TA/instructor to consider while grading.

Please document any known problems with your program. If `gcc` produces any warning or error messages (with the switches: `-std=c99 -pedantic -Wall -Wextra -Wstrict-prototypes -Wno-unused-parameter`), or if `valgrind` reports any problem, please list them, describe what you think they mean, and say whether you think it indicates a problem in your code or if `gcc/valgrind` is just being 'too picky.'

Please, also note approximately how many hours you spent to complete the assignment. This information will not effect your grade, but will be used for future course planning.

2. If you implemented the delete function and phonebook2-extra-credit program, please add a section entitled EXTRA CREDIT. In this section describe what you did for your extra credit work.
3. Under PROGRAM VALIDATION, please describe what you did to test your program or otherwise check that it is behaving correctly. In particular: What inputs did you try? How did you determine if the outputs were correct? Did you try any invalid inputs and make sure that your program doesn't crash? Did you use valgrind when running these tests? (See also the hints of Testing and Debugging given in the instructions to Assignment 5.)
4. Under CERTIFICATION, please answer the following two questions in your README file:
 - (a) Did you use any code/materials other than those provided by or referenced by the instructor or TA in completing this assignment? If so, please cite the source (including its URL if it is online) and the nature and extent of your use of that material.
If such material is prohibited by the directions for this assignment, points may be deducted, but as long as such use is completely disclosed in the README file (including source, nature, and extent of use), it will not be considered to constitute scholastic dishonesty.
 - (b) Did you work with anyone or get help in completing this assignment from anybody other than the instructor or TA? (This includes mentors and tutors.) If so, please describe the nature and extent of the help received.
If such help or collaboration exceeds what is allowed for this assignment according to these directions, points may be deducted, but as long as the help or collaboration is completely disclosed in the README file, it will not be considered to constitute scholastic dishonesty.

Below your answer to those two questions, include the following statement:

```
I certify that except as noted above, the work I submit for
this assignment has been completed solely by me without any
outside help and without looking at any code for solving
this problem other than what is contained in
this document or included in other material/links provided
by the instructor or TA.
```

Below this statement, please write your name and date.

Use svn to commit your final submission by 5pm on the due date.

1. Make sure that you have all of the necessary files and they've all been added to svn.
2. You can continue to make changes, and re-commit as often as you want. The last version committed before 5pm on the due date will be graded.

Permissible Collaboration and Resources. This assignment is to be completed *individually*. You are allowed to discuss general strategies for solving assignments with fellow students or other individuals, but it is no longer a 'general strategy' if the discussion gets to the level of detail of what would be done in actual lines of code found in your programs. In addition, discussions of 'general strategy' should not be taking place while either party is editing their source code. (It is, however, perfectly acceptable to discuss C language constructs and concrete examples of them which are not taken directly from anyone's solution to an assignment.) In addition, you may seek more specific help from mentors and lab tutors in trying to understand why their code doesn't work. Furthermore, you may also seek help and guidance of any kind from the TA and instructor.

For this assignment, you may consult, in addition to any example code or notes provided by the instruction, the following resources:

- the Wikipedia article on AA Trees: http://en.wikipedia.org/wiki/Aa_tree.
- The animation at <http://people.ksp.sk/~kuko/bak/index.html>
- Implementations of or tutorials on AA Trees that were linked directly from the Wikipedia article as of 1:00am, April 3, 2009. (Note: you are not permitted to add additional links to the Wikipedia article!)
- Arne Andersson's paper, 'Balanced Trees Made Simple:' <http://user.it.uu.se/~arnea/abs/simp.html>

You may not, however, consult any C code implementing an AA Tree with string keys.