CS300 – Spring 2019-2020 - Sabancı University

Homework #2 – Phonebook

Due: 18/03/2020, Wednesday, 23:55

Brief description

In this homework, you are going to implement a phonebook which will help the user to make several operations faster such as searching, adding, deleting contacts. For this purpose, you are required to store the contact information in the **Binary Search Tree (BST)** and **AVL** tree and experience the implementation and performance differences between both data structures. Your tree implementations **MUST** be template-based.

The Contacts Input File

We will provide you with input files (phonebooks) which are lists of contacts of different sizes where each line is composed of:

firstName lastName phoneNumber city

There is only ONE space between each field. And, each person has only ONE first name and last name. Therefore, in each line, there are exactly 4 strings. An example input file can be as follows:

```
Elbert Womack +905551674361 Istanbul
Ashley Jepsen +905514534601 Izmir
Carolyn Johnson +905593164303 Mugla
Shirley Mickelson +905543846013 Bursa
Leigh Mathis +905571910433 Mugla
Christopher Cornelius +905521889705 Ankara
Robert Mulcahy +905533468742 Istanbul
Stephanie Turner +905545433234 Bursa
Michael Harrold +905586699074 Izmir
Michael Noseworthy +905578289843 Mugla
Vicky Clark +905525379730 Bursa
Robert Stoll +905572956433 Istanbul
Allen Bonkowski +905521665837 Istanbul
Edward Blauser +905580424941 Bursa
Monica Tarrant +905530429991 Ankara
```

You can assume that the given input file is valid.

The functionalities

There has to be 6 available functions in your program:

1. *SearchContact*
2. *AddContact*
3. *DeleteContact*
4. *InOrderPrintToFile*
5. *PreOrderPrintToFile*

6. *DrawTreeToFile*

AddContact: Adds a new contact for a given firstName, lastName, phoneNumber and city information to both BST and AVL trees. Note that, while inserting into trees, the comparison should be done by alphabetically.
If the given contact info already exists, print a message to warn the user such as "The given contact full name already exists in the database." To decide whether a contact info already exists, a full match must be done for the full name. Note that full name is the concatenation of firstName and lastName by leaving a blank between them. For instance, if firstName is "Ali" and lastName is "Veli", then the full name will be "Ali Veli".

DeleteContact: Given the full contact name (firstName+lastName), your program should remove that contact from the phonebook.

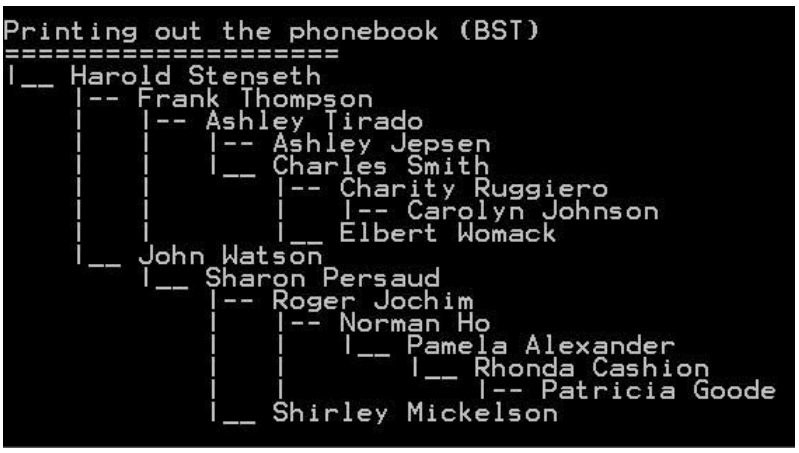
SearchAContact: The search should be performed this way:

- Whenever you enter a full name, your program will display the contact(s) that matches the search full name.
- If a partial string is entered, your program should display the contact(s) whose full name starts with the entered partial word (see sample runs for examples).

InOrderPrintToFile: Print out the phonebook in Pre-order to a file named *phonebookInOrder.txt*. The information that should be printed is the full name, phone number and city, the same as the input file but ordered as *InOrder* sorted.
(Debugging hint: the orderings should be identical).

PreOrderPrintToFile: Print out the phonebook in Pre-order to a file named *phonebookPreOrder.txt*. The information that should be printed is the full name, phone number and city, the same as the input file but ordered as *PreOrder* sorted.

DrawTreeToFile: Prints out both BST and AVL (as shown below) into 2 separate files named as *phonebookTreeBST.txt* and *phonebookTreeAVL.txt*.



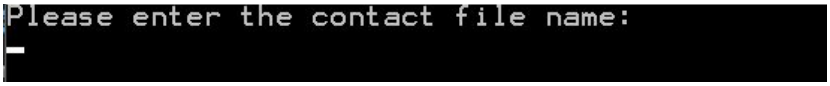
The flow of the program

Once you run your program, it should read the input file (***One of the samples given to you, ex:*** *PhoneBook-sample2.txt*) and load all its contacts into a **BST** and an **AVL** tree. After that, your program should display a menu of functionalities (functions from **1** to **5**, given in sample runs) that it should perform successfully. Once a selection of a function is made, it should be executed for both trees, the BST and the AVL.

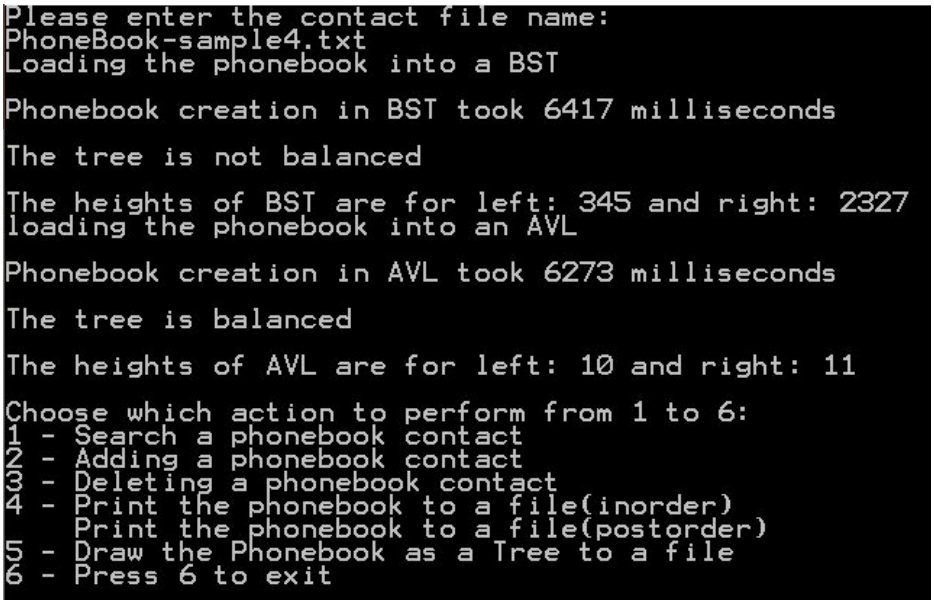
Your program will display the execution times of each operation performed from the list in both trees (The sample runs will explain it thoroughly).

Important: Each node in the tree should contain (Full name, Tel number and City).

Your program should have a list of options to choose which function to run on the phone book, the screenshots below show you how your program should look after running it.



First, it prompts to ask for an input file which is the phonebook *.txt file* that will be provided to you. Then, after entering the **correct** file name, the phonebook should be loaded to both, the **BST** and **AVL** trees. (Don't forget to measure tree creation times for both trees).



From this example, you might have noticed that after loading the phonebook to each tree, it prints the heights of both left and right subtrees for both **BST** & **AVL** trees to show whether the tree is balanced or not. It is **IMPORTANT** to note that your program must redisplay the same menu after running any operation in order to perform another one.

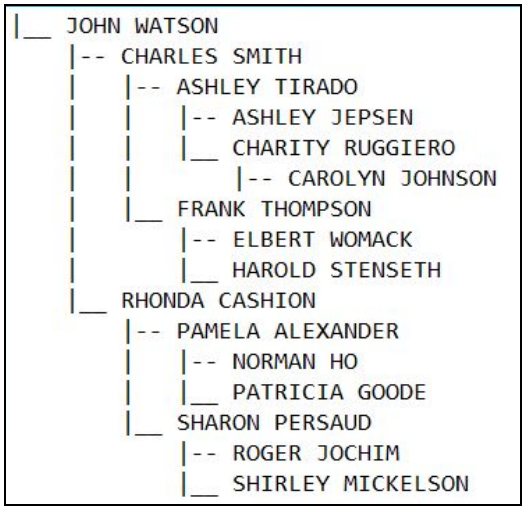
Note: The InOrder and PreOrder functions should be run using the same choice from the menu (choice number 4).

Sample runs

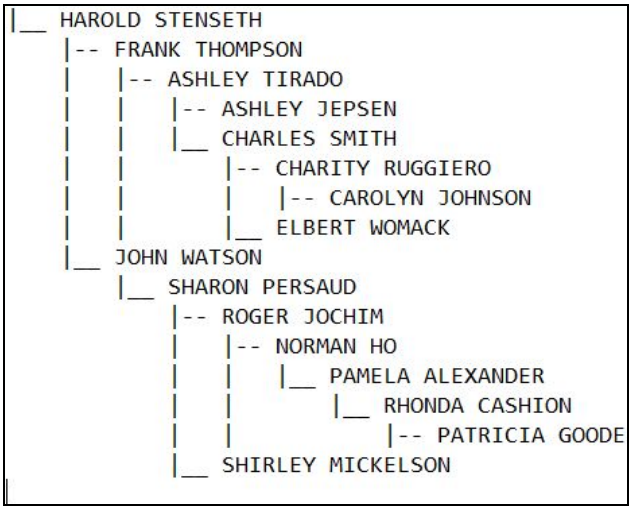
Input file: *PhoneBook-sample(i).txt*
Output files:

- *phonebookInOrder.txt*
- *phonebookPreOrderBST.txt*
- *phonebookPreOrderAVL.txt*
- *phonebookTreeBST.txt*
- *phonebookTreeAVL.txt*

Sample Run 1; *(Printing & Drawing)*
Input file: *PhoneBook-sample1.txt*



phonebookTreeAVL.txt



phonebookTreeBST.txt

```
JOHN WATSON  +905550292913 Izmir
CHARLES SMITH  +905538018232 Mugla
ASHLEY TIRADO  +905555055270 Ankara
ASHLEY JEPSEN  +905514534601 Izmir
CHARITY RUGGIERO  +905567253292 Bursa
CAROLYN JOHNSON  +905593164303 Mugla
FRANK THOMPSON  +905515328944 Istanbul
HAROLD STENSETH  +905547710771 Izmir
RHONDA CASHION  +905531542095 Mugla
PAMELA ALEXANDER  +905597007856 Izmir
NORMAN HO  +905574240361 Ankara
PATRICIA GOODE  +905514483268 Izmir
SHARON PERSAUD  +905593325178 Istanbul
ROGER JOCHIM  +905568022432 Ankara
SHIRLEY MICKELSON  +905543846013 Bursa
```

phonebookPreOrder.txt

```
ASHLEY JEPSEN  +905514534601 Izmir
ASHLEY TIRADO  +905555055270 Ankara
CAROLYN JOHNSON  +905593164303 Mugla
CHARITY RUGGIERO  +905567253292 Bursa
CHARLES SMITH  +905538018232 Mugla
ELBERT WOMACK  +905551674361 Istanbul
FRANK THOMPSON  +905515328944 Istanbul
HAROLD STENSETH  +905547710771 Izmir
JOHN WATSON  +905550292913 Izmir
NORMAN HO  +905574240361 Ankara
PAMELA ALEXANDER  +905597007856 Izmir
PATRICIA GOODE  +905514483268 Izmir
RHONDA CASHION  +905531542095 Mugla
ROGER JOCHIM  +905568022432 Ankara
SHARON PERSAUD  +905593325178 Istanbul
SHIRLEY MICKELSON  +905543846013 Bursa
```

phonebookInOrder.txt

Hint (Drawing): To draw the tree, you can write a recursive function that traverses the tree's levels. It is pretty similar to a pre-order printing function with few additional code lines where you have to draw something according to whether you are in the left subtree or right subtree of each node

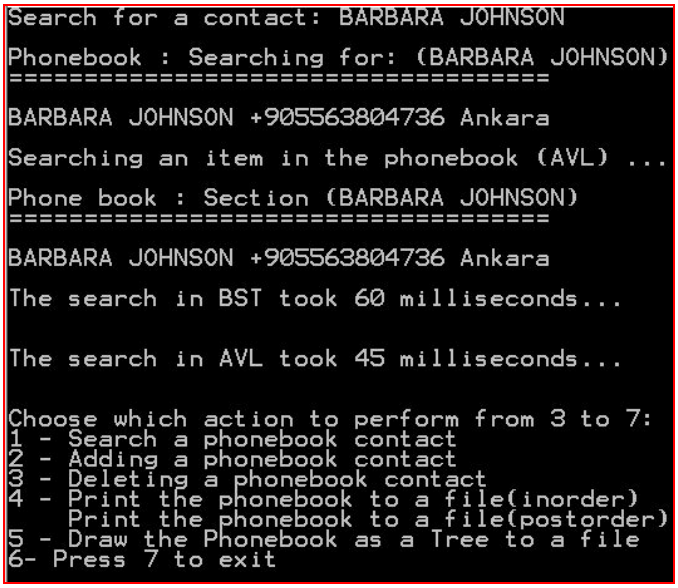
Sample Runs 2 & 3

For Sample runs **2** and **3** you can access their files from the sample runs folder (inside homework2 folder) link since the input sizes are much bigger and there is no room for displaying their outputs in the homework document.

Sample Run 4 *(Searching)*

Input file: *phonk-sample4.txt*

```
Search for a contact: Ada Lenard

Phonebook : Searching for: (ADA LENARD)
===================================

ADA LENARD +905580633219 Izmir

Searching an item in the phonebook (AVL) ...

Phone book : Section (ADA LENARD)
===================================

ADA LENARD +905580633219 Izmir

The search in BST took 64 milliseconds...


The search in AVL took 59 milliseconds...


Choose which action to perform from 3 to 7:
1 - Search a phonebook contact
2 - Adding a phonebook contact
3 - Deleting a phonebook contact
4 - Print the phonebook to a file(inorder)
    Print the phonebook to a file(postorder)
5 - Draw the Phonebook as a Tree to a file
6- Press 7 to exit
```



An example showing the search operation being faster in an AVL than in a BST tree

Another example showing the search operation being faster in an AVL than in a BST tree

Important: You should make sure that your code runs on any random input file of any size, therefore, we would probably use other inputs rather than the samples that we have provided you in the homework folder.

Sample Run 5 *(Deletion)*

As you will notice in the figure below, the deletion in the AVL tree took less time than the deletion in the BST, it is about 500 times faster. However, when the maximum height in the BST is close to the maximum height of the AVL tree, then there will be cases where the deletion in BST is faster than in AVL. Please **note** that you should print a message to indicate that the deletion "**terminated successfully**" in case the contact to be deleted exists in the phonebook, otherwise, it should print "**Not found**".

```
Choose which action to perform from 3 to 7:
1 - Search a phonebook contact
2 - Adding a phonebook contact
3 - Deleting a phonebook contact
4 - Print the phonebook to a file(inorder)
    Print the phonebook to a file(postorder)
5 - Draw the Phonebook as a Tree to a file
6 - Press 6 to exit

3

Deleting an item from the phonebook ...
====================
Enter the fullname of the contact to be deleted:
HEATHER SCHARRINGHAUSEN
Deleted succesfully...
Deletion from Binary Tree took 150450227 nanoseconds...
Deletion from AVL Tree took 289717 nanoseconds...

Choose which action to perform from 3 to 7:
1 - Search a phonebook contact
2 - Adding a phonebook contact
3 - Deleting a phonebook contact
4 - Print the phonebook to a file(inorder)
    Print the phonebook to a file(postorder)
5 - Draw the Phonebook as a Tree to a file
6 - Press 6 to exit
```

Sample Run 6 *(Insertion)*

```
Adding an item to the phonebook (BST) ...
====================
Enter the information of the contact to be added:
Name: Gulsen Demiroz

Tel: +905513324477

City: Istanbul

Contact has been added successfully to the BST

Adding an item to the phonebook (AVL) ...
====================

Contact has been added successfully to the AVL tree


Adding a contact to the Binary Tree took 9331 nanoseconds...
 Adding a contact to the AVL tree took 22860 nanoseconds...

Choose which action to perform from 3 to 7:
1 - Search a phonebook contact
2 - Adding a phonebook contact
3 - Deleting a phonebook contact
4 - Print the phonebook to a file(inorder)
    Print the phonebook to a file(postorder)
5 - Draw the Phonebook as a Tree to a file
6 - Press 6 to exit
```

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all homeworks, unless otherwise noted.

How to get help?

You may ask questions to TAs (Teaching Assistants) of CS300. Office hours of TAs can be found here. Recitations will partially be dedicated to clarify the issues related to homework, so it is to your benefit to attend recitations.

What and Where to Submit

Please see the detailed instructions below/in the next page. The submission steps will get natural/easy for later homeworks.

Grading

Careful about the semi-automatic grading: Your programs will be graded using a semi-automated system. Therefore, you should follow the guidelines about input and output order; moreover, you should also use the exact same prompts as given in the Sample Runs. Otherwise semi-automated grading process will fail for your homework, and you may get a zero, or in the best scenario you will lose points.

Grading:

- **We will grade your homeworks during the demo session. Each one of you must show that your code is running as expected and explain several parts of your code if necessary. Wait for an announcement for the demo scheduling.**
- Late penalty is 10% off the full grade and only one late day is allowed.
- **Having a correct program is necessary, but not sufficient to get the full grade. Comments, indentation, meaningful and understandable identifier names, informative introduction and prompts, and especially proper use of required functions, unnecessarily long program (which is bad) and unnecessary code duplications will also affect your grade.**
- Please submit your own work only (even if it is not working). It is really easy to find out "similar" programs!
- For detailed rules and course policy on plagiarism, please check out http://myweb.sabanciuniv.edu/gulsend/courses/cs201/plagiarism/

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Grade objections: Since we will grade your homeworks with a demo session, there will be very likely no further objection to your grade once determined during the demo.

What and where to submit (IMPORTANT)

Submission guidelines are below. Most parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Add your name to the program: It is a good practice to write your name and last name somewhere in the beginning program (as a comment line of course).

Name your submission file:

- Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.
- Name your cpp file that contains your program as follows.

  "SUCourseUserName_yourLastname_yourName_HWnumber.cpp"
- Your SUCourse user name is actually your SUNet username which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

  cago_ozbugsizkodyazaroglu_caglayan_hw2.cpp
- Do not add any other character or phrase to the file name.
- Make sure that this file is the latest version of your homework program.
- You need to submit ALL .cpp and .h files including the data structure files in addition to your main.cpp in your VS solution.
- The name of the main cpp file should be as follows.

  "SUCourseUserName_yourLastname_yourName_HWnumber.cpp"

  For example zubosman_Osmanoglu_Zubeyir_hw3.cpp is a valid name, but hw2_hoz_HasanOz.cpp, HasanOzHoz.cpp are NOT valid names.

Submission:

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Anes Abdennebi, Artun Sarıoğlu and Gülşen Demiröz