CS300 – Fall 2023-2024 - Sabancı University

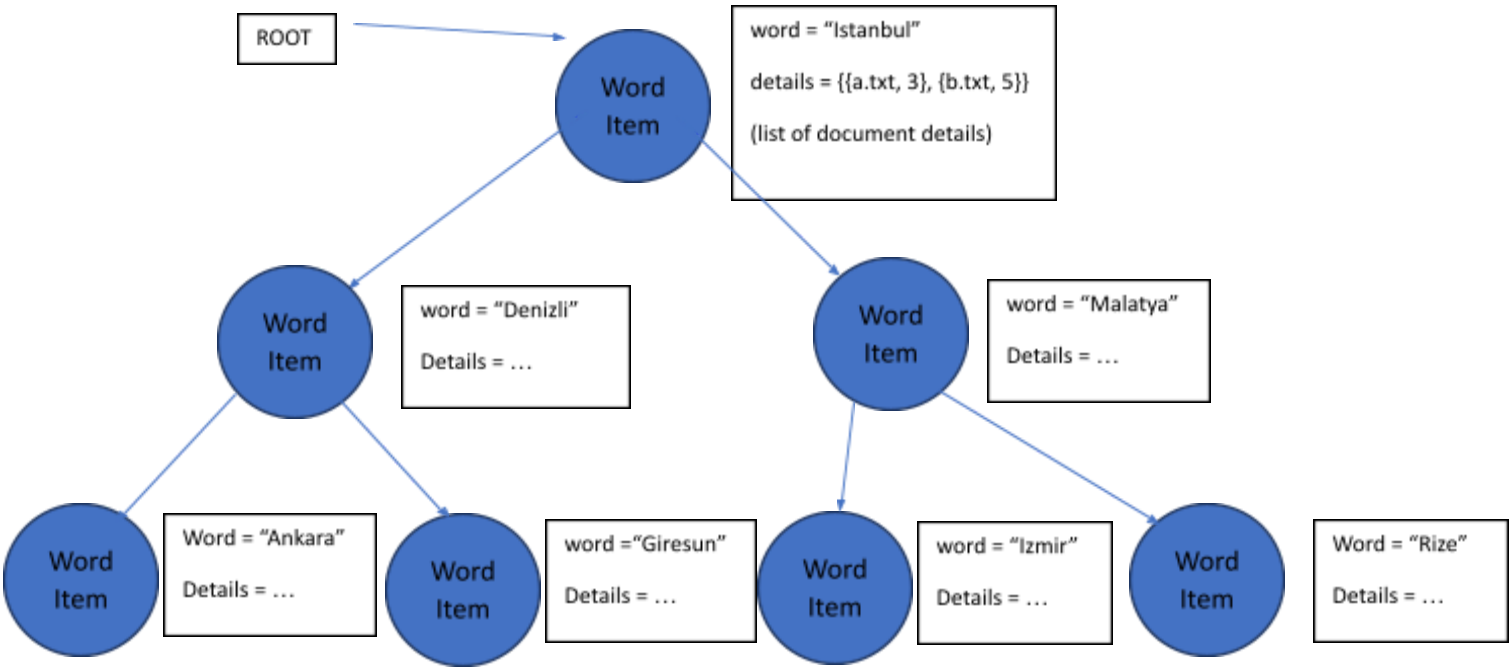
Homework #2 – Search Engine

Due November 24 Friday at 22:00

Brief Description

In this homework, you will write a search engine. The search engines in real life, search hundreds of millions web pages to see if they have the words that you have typed, and they can do this for thousands of users at a given time. In order to do these searches really fast, search engines such as Google do a lot of what we call preprocessing of the pages they search; that is, they transform the contents of a web page (which for the purposes of this homework, we will assume to consist of only strings) into a structure that can be searched very fast.

In this homework, you are going to preprocess the documents provided to you. For each unique word, you will insert a node into your **AVL Search Tree**. Of course you will also keep track of the details such as the document name which the word appears in and the number of times the word appears in each document. So, you need to implement a templated **AVL Search Tree** first.



Program Flow

You need to get the number of documents that you need to preprocess first. Then, after getting the names of all documents from the console, you need to preprocess all the text in the documents. Only the alphabetical characters are considered, and the unique words need to be case insensitive (for example; “izmir” and “Izmir” are the same). The rest (such as digits or punctuation characters) will be ignored. For each unique word appearing in the text, you need to insert a new node into the tree. If the node is already there, you need to update the node with the information about this document. For example; you have preprocessed the “a.txt” document first and there is only one “the” word in this document. You need to insert “the” word into the **AVL Search Tree**. Then, while preprocessing the “b.txt” document, “the” word appears 4 times. So, you need to add this information ({“b.txt”, 4}) into the existing node in your BST and the node of “the” word becomes {{“a.txt”, 1}, {“b.txt”, 4}}.

After you preprocess all the documents, you need to get a query from the console which consists of a line of strings (HINT: You may use getline(cin, line)). This line might consist of more than one word. Then, you need to output the document names in which all words in the query appear, including the number of times that each word appears in that document.

Hint

```
struct DocumentItem {
     string documentName;
     int count;
};

struct WordItem {
     string word;
     // List of DocumentItem's. In order to keep the documents
     //you can again use the BST that you are going to
implement.
};

template <class Key, class Value>
class AVLSearchTree
{
   …
};
```

and then when you create a **AVL Search Tree** object it looks like this:

```
AVLSearchTree<string, WordItem *> myTree;
```

Sample Runs

The example text files that we will use in the sample runs are as follows:

a.txt

```
Sabanci sabanci cs CS FENS Engineering Computer
```

b.txt

```
Homeworks are so fun. WoW, What a story!!!
Sabanci is so cool. I love 2 CS CS CS CS
```

Sample Run

```
Enter number of input files: 2
Enter 1. file name: a.txt
Enter 2. file name: b.txt

Enter queried words in one line: Sabanci
in Document a.txt, sabanci found 2 times.
in Document b.txt, sabanci found 1 times.

Enter queried words in one line: cs sabanci
in Document a.txt, cs found 2 times, sabanci found 2 times.
in Document b.txt, cs found 4 times, sabanci found 1 times.

Enter queried words in one line: sabanci fens
in Document a.txt, fens found 1 times, sabanci found 2 times.

Enter queried words in one line: fun computer Engineer
No document contains the given query

Enter queried words in one line: REMOVE sabanci
sabanci has been REMOVED

Enter queried words in one line: Sabanci
No document contains the given query

Enter queried words in one line: ENDOFINPUT
```

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 at SUCourse. 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

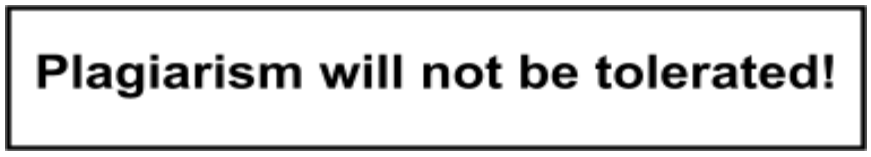
Please see the detailed instructions below/in the next page. The submission steps will get natural/easy for later homeworks.

Grading and Objections

Your programs 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 the grading process will fail for your homework, and you may get a zero, or in the best scenario you will lose points.

Grading:

- ❑ 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 programs (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 "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, dot 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.

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!
Gülşen Demiröz