CS 202, Spring 2016
Homework Assignment 4

Assigned: April 11, 2016
Due: 23:59, April 24, 2016

Question 1 (30 points)

Assume that we have the following balanced-searched tree implementations:

a) AVL tree

b) 2-3 tree

c) 2-3-4 tree

Starting with an empty balanced search tree, we would like to insert the following keys into the tree in the given order:

```
23 36 52 45 61 78 57 29
```

and then, delete the keys `45 57 61` (in the given order) from the tree. Note: while deleting an internal node, its inorder successor should be used as the substitute if needed.

Show the underlying tree for the AVL, 2-3, and 2-3-4 implementations after *each* insertion and deletion.

Question 2 (10 points)

Assume that we have a hash table with size 13 (index range is $0 \ldots 12$), and we use the `mod` operator as a hash function to map a given numeric key into this hash table. Draw the hash tables after the insertion of the keys

```
15 13 9 12 28 11 25 24
```

in the given order for each of the following methods:

a) open addressing with linear probing,

b) open addressing with quadratic probing,

c) separate chaining.

Question 3 (60 points) (Programming Assignment)

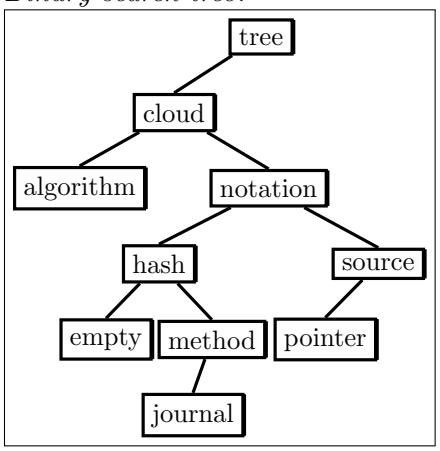
In this question, you are asked to write a C++ program for querying a dictionary of English words. The queries will be run on a binary search tree-based and a 2-3 tree-based implementation of the dictionary. Your program will first read the contents of the dictionary from an input file, and build the search trees. The file will contain an unsorted list of words with each word given in a separate line. Your program MUST build the trees by inserting these words in the order that they appear in the file.

For example, for the given dictionary below, the following trees are constructed. You can assume that all words are unique.

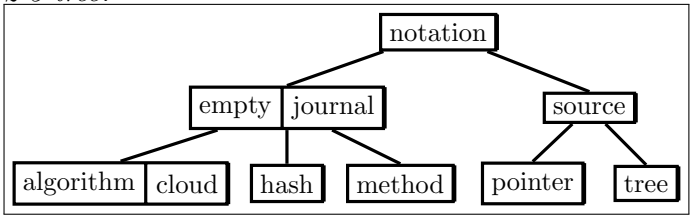
Dictionary:

tree
cloud
notation
source
hash
empty
pointer
method
journal
algorithm

Binary search tree:



2-3 tree:



After building the trees, your program will read another input file that contains the query words (each word in a separate line), search each of these words in the dictionary, and generate an output file consisting of the search statistics. Each line in the output file must present the search result for the corresponding word in the query file. The line should show the word, whether it is found in the dictionary or not (1 or 0), and the number of key (string) comparisons made during the search. You should also show the maximum and average number of comparisons made.

For example, for the given query words below, the following results are obtained. You can assume that the comparisons are case insensitive.

Queries:

density
source
pointer
parse
journal
algorithm
name
cloud

Binary search tree:

density	0	5
source	1	4
pointer	1	5
parse	0	5
journal	1	6
algorithm	1	3
name	0	5
cloud	1	2

# of queries: 8
Maximum # of comparisons: 6
Average # of comparisons: 4.38

2-3 tree:

density	0	4
source	1	2
pointer	1	3
parse	0	4
journal	1	3
algorithm	1	3
name	0	4
cloud	1	4

# of queries: 8
Maximum # of comparisons: 4
Average # of comparisons: 3.375

Use the following interfaces for implementing the dictionary using binary search trees and 2-3 trees. You can define additional member functions and data members but do not modify the definitions of the given member functions. An example `main` function for using these classes is also given below.

The class definitions must be in the files named `DictionaryBST.h` and `Dictionary23Tree.h`, respectively, and the class implementations must be in the files `DictionaryBST.cpp` and `Dictionary23Tree.cpp`, respectively. You can use additional files in your implementation. You are expected to test your code by writing a driver `main` function but do not submit your `main` function as part of your code.

```
//Binary search tree implementation for the dictionary
class DictionaryBST {
    public:
        //Constructor that constructs the dictionary from the input file
        DictionaryBST( string dictionaryFile );

        //Destructor
        ~DictionaryBST();

        //Inserts the given word into the dictionary
        void insert( string word );

        //Searches the given word in the dictionary, and returns the number
        //of comparisons made and whether the word is found or not
        void search( string word, int& numComparisons, bool& found );

        //Searches all words in the query file in the dictionary, and
        //summarizes the results in the output file
        void search( string queryFile, string outputFile );
        ...
};
```

```
//2-3 tree implementation for the dictionary
class Dictionary23Tree {
    public:
        //Constructor that constructs the dictionary from the input file
        Dictionary23Tree( string dictionaryFile );

        //Destructor
        ~Dictionary23Tree();

        //Inserts the given word into the dictionary
        void insert( string word );

        //Searches the given word in the dictionary, and returns the number
        //of comparisons made and whether the word is found or not
        void search( string word, int& numComparisons, bool& found );

        //Searches all words in the query file in the dictionary, and
        //summarizes the results in the output file
        void search( string queryFile, string outputFile );
        ...
};
```

```
int main() {

    DictionaryBST dictBST( "dictionary.txt" );

    dictBST.search( "query.txt", "outputBST.txt" );

    Dictionary23Tree dict23Tree( "dictionary.txt" );

    dict23Tree.search( "query.txt", "output23Tree.txt" );

    return 0;
}
```

Submission:

- This assignment is due by 23:59 on Sunday, April 24, 2016. You should upload your solutions using the online submission form at `http://www.cs.bilkent.edu.tr/~saksoy/courses/cs202-Spring2015/upload.html`. You should upload a single zip file that contains:
  - Your answers to questions 1 and 2 in a file named `firstname_lastname_hw4.pdf`. Do NOT send photos of your solution, type/draw if possible, scane if not.
  - Your code files for question 3. The names of your code files must be `DictionaryBST.h`, `Dictionary23Tree.h`, `DictionaryBST.cpp`, and `Dictionary23Tree.cpp`. You can also write additional files but do not include your test `main` function in any of these files.
  - Do not put any unnecessary files such as the auxiliary files generated from your favorite IDE. Be careful to avoid using any OS dependent utilities.
  - Do not forget to write your id, name, section, assignment number and any other information relevant to your program in the beginning of every file that you are submitting.
  - Do not forget to write comments at important parts of your code.
  - Any violation of these may cause a significant loss from your grade.
- **For this assignment, you must use your own implementations of binary search trees and 2-3 trees. You can adapt the codes and pseudocodes in the Carrano book. However, you are NOT allowed to use any existing codes from other sources (including the codes given in other textbooks, found on the internet, and belonging to your classmates). Furthermore, you are NOT allowed to use any data structure or function from the C++ standard template library (STL).**
- Although you may use any platform or any operating system to implement your solutions, your code should work in a Linux environment (specifically using the `g++` compiler). We will compile your programs with the `g++` compiler and test your codes in a Linux environment. Thus, you may lose a significant amount of points, if your C++ code does not compile or execute in a Linux environment.
- Keep all the files before you receive your grade.
- This homework will be graded by your TA Ali Yesilyaprak (aliyesilyaprak at gmail.com). Thus, you may ask your homework related questions directly to him.