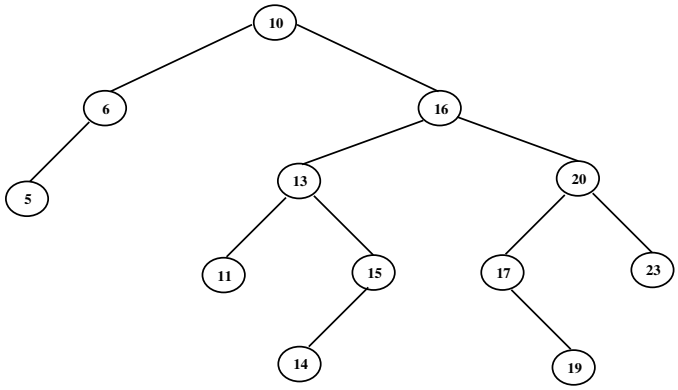
Assignment 1

Computer Science 2300, Fall 2005
Due: Sep 19, 2005

1 Theory (40 points)

1. (**5 points**) Kleinberg and Tardos, problem 2.1
2. (**5 points**) Kleinberg and Tardos, problem 2.2
3. (**5 points**) Kleinberg and Tardos, problem 2.3
4. (**10 points**) Kleinberg and Tardos, problem 2.8
5. (**5 points**) Delete the node with key-value 16 from the following binary search tree, using the algorithm discussed in class.



6. Suppose you are given the following set of 15 key-values: 21, 97, 56, 32, 1, 76, 29, 43, 17, 84, 63, 81, 4, 28, 49.
   (a) (**5 points**) In what order should these keys be inserted to *maximize* the height of the binary search tree? Explain.
   (b) (**5 points**) Draw the binary search tree which has these key-values as nodes and has *minimum* height.

2 Code (40 points)

2.1 Description

For the coding portion of this assignment, you are going to create a system for managing a collection of names, addresses, and phone numbers. You will be given the front-end for entering and retrieving this contact information, as well as a `Contact` class to store information for a single contact. You are responsible for creating a binary search tree class to store a collection of `Contact` objects.

2.2 Implementation

You are to create a class called `BinarySearchTree` that exposes the following API:

Method	Description
`BinarySearchTree()`	Default Constructor
`~BinarySearchTree()`	Destructor (be sure to free any memory you allocated)
`void insert(Contact contact)`	Inserts a new node into the tree using the given `Contact` object as the node's value. If a matching node already exists in the tree, replace the existing value with the given value.
`void remove(string name)`	Removes the specified contact's node from the tree
`bool contains(string name)`	Returns true if the tree contains contact information for the given contact name
`int size()`	Returns the total number of nodes in the tree
`Contact get(string name)`	Returns the `Contact` object contained in the search tree for the given contact name
`void printInOrder()`	Prints a list of all contacts in the binary tree in ascending alphabetical order by name

You are allowed to add other helper functions to your `BinarySearchTree` class as needed, but they should be declared Private.

2.3 Deliverables

Turn in `BinarySearchTree.cpp` and `BinarySearchTree.h`

2.4 Grading

We will run your code on a number of example cases, to see how it performs on the various operations you are asked to implement, such as insertion, deletion, and search. Your grade will be based on the success or failure of your code on these example cases.

2.5 Hints

- The `Contact` class has overloaded comparison operators such that you can directly compare (case insensitively) a `Contact` object to a string containing a contact's name.
- The `Contact` class also has overloaded the `<<` operator so that `cout << c << endl`, where `c` is an instance of a `Contact` object, will output properly formatted contact information for `c`.

3 Submission Guidelines

3.1 Theory

You must turn in a copy of your answers to the theory questions at the beginning of class on Sep 19, 2005.

3.2 Code

You must submit your code by 11:59 pm on Sep 19, 2005. Your submission must include all of your code. Here's how to submit:

1. **Copy your files to RCS (RPI's unix system).** There are many ways to copy files from your local machine to RCS (contact the help desk if you do not know how). If you use FTP to copy your files, you must make sure you transfer the files in text format. If you transfer the files in binary format, your submission will be corrupted and you can expect a VERY poor grade.

2. **Create a tar archive containing the files. The name of the archive file should be your RCS user ID.** To create a tar archive, use the tar command with two flags `c` (create) and `f` (file), followed by the name of the tar file and followed by a list of all the files you want to include. For example, user `brahms` would type

   ```
   tar cf brahms.tar BinarySearchTree.cpp BinarySearchTree.h
   ```

3. **Copy the tar archive to the correct (!) submission directory.** The submission directory is `/dept/cs/cs230/assignment1/`. To copy your file to this directory, use the following command:

   ```
   cp brahms.tar /dept/cs/cs230/assignment1/
   ```

   You'll find that you cannot over-write your submission file after it has been copied. If you need to update your submission, use the following numbering system:

   First submission: `userid.tar`

   Second submission: `userid-2.tar`

   Third submission: `userid-3.tar`

   ...

   Only your last submission will be graded. The time stamp on the last tar file (check using the command `ls -l`) will determine the submission time of your project.

3.3 Help

Help is available through recitations, TA office hours, and emailing the TAs. Use these resources if you need to.

3.4 Final Warning

Proper submission is entirely **your responsibility**. Contact your TA if you have any doubts whatsoever about yoursubmission. Be sure to keep copies of your files, including your tar file, and **do not change them after submitting**. Don't even copy them. After grades are posted, you have exactly one week to resolve all problems. After that week is up, all grades are final. **Remember: if you ask for your project to be regraded, your grade may go up OR down.**