HW5: Programming Question A and Written Homework (except problems 12 thru 17), due 11:59, Mon. Dec. 3, 2012
HW6: Programming Questions B and C, and Written problems 12 thru 17, due 11:59, Mon. Dec. 10, 2012

Your fifth and sixth assignment includes both a programming portion and a written portion. The programming portion should consist of a separate file for each assignment, hw05.cpp and hw06.cpp. The written portion should consist of a separate file for each assignment, hw05written and hw06written, saved in a standard document format (.txt, .doc, .htm., or .pdf). Be sure to include your name at the beginning of each file! *You must hand in both files via MyPoly.*

Programming Part:

A) Add the following methods to the `BinarySearchTree` class. Each method should be solved recursively.

1. Create a method called `CountNodesWithOneChild` that returns the number of nodes in the tree having only one child.

2. Create a method called `negateTree` that changes every value in the tree to its negated value. (The left and right child should be swapped, thus maintaining the binary search tree)

```
  a
/   \
b    c
becomes
  -a
 /   \
-c   -b
```

3. Create a recursive method called `max_sum_root_to_leaf` that computes the maximum sum of a root to leaf path. Your method must be efficient. What is the running time of your algorithm?

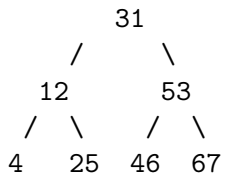
   e.g.

```
      6
    /  \
   3    8
 /  \
2    4
      \
       5
```

   This tree has `3` root to leaf paths. The maximum sum is 18.

B) Add a method to the binary search tree class that lists the nodes of a binary tree in level-order (It should first list the root, then the nodes at depth 1, then the nodes at depth 2, and so on). Your algorithm should have a worst case running time of $O(n)$, where n is the number of nodes in the tree. Explain how your algorithm fulfills this requirement. *Hint*: The ADT Queue might be helpful. [1]

e.g. For the following tree:



Printing the nodes of this tree in a level order would output: $31, 12, 53, 4, 25, 46, 67$.

C)

Write the code to determine how to go from one subway station to another in the smallest number of hops (i.e. travel through the smallest number of subway stops.)

To implement this, you have two options:

1. Do not use any code you have previously written, and perform the following steps:

   - Create three objects: the graph of type `vector< list < int > >`, an object to associate each `stop_id` with an index in the vector, and an object to associate each index with a `stop_id`. These three objects need to be given values (somewhat) simultaneously.
     - Create a mapping between the `stop_id` and the index into the `vector` . To do this efficiently, you will use a hash table that stores `<key, data>` pairs. There are two C++ hash tables that store `<key, data>` pairs. For your hash table, your `key` will be the `stop_id` and the `data` will be the index into the `vector`.

       Note: I used the hash table in the `hash_map` container:

       `http://www.sgi.com/tech/stl/hash_map.html`

       The other is C++ hash table is in the `unordered_map` container: [2]

       `http://www.cplusplus.com/reference/unordered_map/unordered_map/`

       You may also use the separate chaining hashing code that was presented in class.
     - Create a vector that maps the array index to the `stop_id`. This can be done by simply using a `vector<string>`.
     - Create a graph, `g`, (i.e. an adjacency list) of type `vector< list < int > >`. The are two differences between creating the adjacency list for this assignment and the adjacency list created in the previous assignment. In this assignment, the adjacency information is stored in a `vector< list < int > >` instead of `vector < train_stop_data >`. For this assignment, you will store the index into the `vector`, `g`, instead of the storing the `stop_id`.

[1] Level-order traversal of a tree is also called breadth-first traversal.

[2] Both these containers have constant average time and linear worst case time.

- Modify the shortest path algorithm, `shortestpaths`, given in class so that it computes the shortest path to travel from one `stop_id` to another `stop_id`. To do this, you will need to change the parameters of the algorithm, `shortestpaths` (in addition to modifying the algorithm). Remember to print out the `stop_id`'s and not the indexes into the `vector`, `g`.

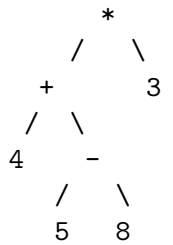
Note: Some additional comments will be given in the hints file.

2. (10 points extra credit) Instead of following the steps outlined in the previous option, you will add another item to the MTA menu to tell the user how to get from one `stop_id` to another `stop_id`.

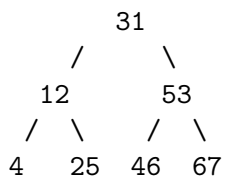
   To get this extra credit, your code must run, be cleanly written, and be efficient (or at least not too inefficient). For 2 additional extra credit points, allow the user to enter two locations (the locations specified by longitude and latitude), then tell the user which subway stop is closest to their location and how to get to the subway stop closest to their destination, while traveling through the fewest subway stops.

Written Part:

1. Write the pseudo code for:
   - Programming problem A. For each method, write the 3 to 6 steps needed to solve this problem.
   - Programming problem B. Write the 4 to 8 steps needed to solve this problem.
2. For the `BinaryTree` class, what is the running time of:
   - `printPreOrder( )` method
   - `size( )` method
   - `height( )` method
   - `isEmpty( )` method
   - `makeEmpty( )` method
   - `merge(x, t1, t2 )` method
3. What is the maximum number of nodes in a binary tree of height h?
4. What is the minimum number of nodes in a binary tree of height h?
5. Given the implementation of the binary search tree presented in class, what is the best order to insert the following numbers $\{0, 1, 2, 3, 4, 5, 6, 7\}$ so that the tree has minimal height? Show the tree that would be created if they were added in that order.
6. (a) Show the result of inserting $(2, 1, 4, 5, 8, 3, 6, 7)$ into an initially empty binary search tree.
   (b) What is the output of an inorder traversal of your tree obtained in (a).
   (c) What is the output of a preorder traversal of your tree obtained in (a).
   (d) What is the output of a postorder traversal of your tree obtained in (a).
7. Show the result of inserting $(2, 1, 4, 5, 8, 3, 6, 7, 9)$ into an initially empty Red-Black tree as described in class. **Include** the color of each node. Show the tree before and after each violation, and state which violation has occurred (i.e. case 1, case 2, or case 3.)
8. Print the nodes of the following expression tree in post order:



9. Given the following binary search tree, remove 31 from the tree using the algorithm from class.



10. Write the code needed to perform the method `rightRotateRecolor` for the `RedBlackNode` class. The method has the following prototype:

```
template <class Comparable>
void RedBlackTree<Comparable>::rightRotateRecolor( Node * & k2 )
```

11. For a hash function `H(x) = x mod 10`, and a fixed table size of `10`,

    (a) if the collision strategy was `linear probing`, what would the hash table look like
    - after inserting `4371, 1323, 6173, 4199, 4344, 9679`
    - and then removing `6173`
    - and then inserting `3324`

    (b) if the collision strategy was `separate chaining` what would the hash table look like
    - after inserting `4371, 1323, 6173, 4199, 4344, 9679`
    - and then removing `6173`
    - and then inserting `3324`

12. For the graph in the file `Homework 6 Graphs`, show the adjacency list representation of the graph:

```
Shanghai :-> Tokyo -> Sydney

Tokyo    :-> Hong Kong -> Sydney -> Los Angeles
.
.
.
```

13. Illustrate the execution of the single source shortest path algorithm for an unweighted graph (breadth first search) on the (undirected) graph in the file `Homework 6 Graphs`, with source vertex `Shanghai`.

    At each phase, show the node being visited in that phase, and the relevant data at the end of the phase:

    - the distance to each vertex that has been discovered (including those already visited)
    - the predecessor of each discovered node on a shortest path from the source
    - the queue of nodes that have been discovered but not yet visited

    The answer at the end of the first phase is shown in the table below. It indicates that at the end of the first phase, we have visited `Shanghai` and have discovered `Tokyo` and `Syndey`, each with estimated distance 1 and with predecessor `Shanghai`. Complete the table. (continue till all the nodes have been removed from the queue.)

```
phase distance/predecessor              visiting   queue
        Sh   T   L   H   Sy   A   R                (front at queue on the left)
-------------------------------------------------------------------------------
init    0/-                                          Sh
-------------------------------------------------------------------------------
1       0/- 1/Sh    1/Sh 1/Sh              Sh          T Sy H
-------------------------------------------------------------------------------
2
-------------------------------------------------------------------------------
.
.
.
```

    Use the chart you have created to determine a shortest path from `Shanghai` to `Rarotonga`.

14. If we implemented a `priority queue` using an `unordered vector`, `v`, where inserting a new item `x` is performed by the code `v.push_back(x)`, how long would it take to perform the following operations of a priority queue? Write your answer using Big-Oh notation.

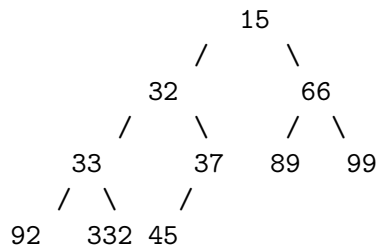
    - `insert(x);` // Average case time:_____
    - `deleteMin();` // Average case time:_____
    - `findMin();` // Average case time:_____

15. Consider the following array representation of a binary heap:
    `[sentinel, 10, 38, 22, 88, 40, 77, 70, 100, 999, 42, 50, 81]`

    (a) Show the tree representation of the binary heap.

    (b) Insert 17 into the binary heap; show both the tree representation and the array representation after 17 has been inserted.

16. Consider the following tree representation of a binary heap:



    (a) Show the array representation.

    (b) Show what happens when the root is removed by giving the tree representation of the binary heap.

17. Illustrate the execution of Dijkstras algorithm on the weighted (undirected) graph in the file `Homework 6 Graphs`, with source vertex `Shanghai`:

Show all the steps using the following chart:

phase	distance/predecessor						visiting	discovered
	Sh	T	L	H	Sy	A	R	
init	0	inf	inf	inf	inf	inf	inf	Sh
1	0	4/Sh	inf	3/Sh	12/Sh	inf	inf	Sh	T Sy H
2								
.								
.								
.								

i.e. at each phase, show:

- which phase your are in
- the node being visited in that phase
- the current estimate of distance to each vertex (via nodes already visited)
- the predecessor of the node (on a shortest path via nodes already visited, i.e. the current estimate of the predecessor)
- all the nodes that have been discovered, but not yet visited.

Continue the chart until all the nodes have been visited.

Using the chart you created, **determine** a shortest weighted path from `Shanghai` to `Rarotonga`.