CS2134 HOMEWORK 9
Spring 2016
Due* 11:00 p.m. on Wed. April 27, 2016

Be sure to include your name at the beginning of each file! Assignment 9 include a programming portion and a written part. The programming portion must compile and consist of a single file (hw09.cpp). The written portion should consist of a single file (hw09written) in a .pdf format. Be sure to include your name at the beginning of each file! You must hand in the file via NYU Classes.

You must hand in both files via NYU Classes.

1. Implement a hash table where collisions are resolved by linear probing. You will implement the methods `find`, `insert`, and `rehash` for the class below. Your class will `rehash` when the load factor is greater or equal to 0.5. The hash function, `hf`, will be created when you instantiate a member of this class. But `hf` will returna positive integer whose range is much larger than the table size. The book says, "you will need to perform a final `mod` operation internally after the" hash function.

```
template< class HashedObj >
class HashTable
{
  public:
     explicit HashTable( int size = 101 ):currentSize(0){ array.resize(size); }
     std::hash<HashedObj> hf; // create a hash function object
     bool find( const HashedObj & x ) const;
     void makeEmpty( );
     bool insert( const HashedObj & x );
     bool remove( const HashedObj & x);
     enum EntryType { ACTIVE, EMPTY, DELETED };
 private:
     struct HashEntry
     {
          HashedObj element;
          EntryType info;
          HashEntry( const HashedObj & e = HashedObj(), EntryType i = EMPTY )
          : element( e), info(i) {}
     };
     vector<HashEntry> array;
     int currentSize;
     void rehash( );
 };
```

*10% extra credit will be given if you turn this assignment in on Tuesday April 26, 2016 at 11:00 p.m.

2. Create an *adjacency list* for the graph of the NYC subway station. You will consider a subway station `sa` adjacent to subway station `sb` if there is a subway train that goes from `sa` directly to `sb` without going through any other stops, or you can transfer from `sa` to `sb`. You find what subway stations are adjacent to other subway stations by using the information in the files `transfers.txt` and MTA train stop data. The file `transfers.txt` is a bit confusing! Here are some of the entries:

```
from_stop_id,to_stop_id,transfer_type,min_transfer_time
101,101,2,180
103,103,2,180
104,104,2,180
106,106,2,180
107,107,2,180
108,108,2,180
109,109,2,180
110,110,2,180
111,111,2,180
112,112,2,180
112,A09,2,180
113,113,2,180
...
```

In the file transfers.txt, observe you can transfer from subway station 112 to subway station $A09$. Thus $A09$ is adjacent to (a neighbor of) 112. You will create an adjacency list as an `unordered_map<string, list<string>>` instead of a `vector<list<string>>`. In the next homework assignment, you will use this adjacency list in the shortest path algorithm.

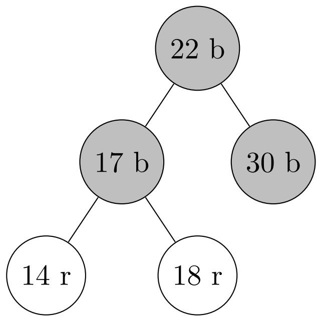
Written Part

1. Using big-oh notation, what is the worst case running time for your algorithm to create the adjacency list?

2. First add `12`, then `50`, and finally `44`, to this red-black tree (i.e. first insert `12` and perform any operations necessary (using only the algorithm presented in class) so the tree is again a red-black tree, then insert `50` and perform any necessary operations (using only the algorithm presented in class) so the tree is again a red-black tree, and then insert `44`and perform any operations necessary (using only the algorithm presented in class) so the tree is again a red-black tree.)

   Show the following:

   - Show the tree after inserting a value. If a violation occurred, state what the violation was (i.e. case 1, 2 or 3).
   - If the tree is not a red-black tree, use the algorithm we used in class to turn it into a red-black tree. **Show each step.**



3. **Do not turn in.** Please do this on your own. I will expect you to know how to do this for the exam. You can check your solution at `https://www.cs.usfca.edu/~galles/visualization/RedBlack.html`. Show the result of inserting `(15, 4, 2, 8, 16, 9, 5, 10)` 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).

4. 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 )
```

5. When inserting an item into a Red-Black tree, what is the maximum number of pointer changes needed to adjust the tree? Explain your answer.

6. If your hash table size, n, was equal to 20 (or 40, or 2000), why might the following hash function *not* be a good idea: $h(k) = 4k \bmod n$?

7. For the following questions: Let the hash function be `H(x) = x mod M`, where the table size (`M`) is initially 5.

   (a) if the collision strategy was `linear probing`, what would the hash table look like
   - after inserting `4371, 6173`
   - then removing `6173`
   - then inserting `3327` and `26`
   - after resizing[1] to a table size of `M = 11`
   - then inserting `4199, 4340, 9679, 1323`

   (b) if the collision strategy was `separate chaining`, what would the hash table look like
   - after inserting `4371, 1323, 6173, 4199, 4344, 9679`
   - then removing `6173`
   - then inserting `3324`
   - then resizing[2] to a table size of `M = 11`
   - inserting `21`

[1]where all the items must be inserted into the table by rehashing
[2]using the algorithm on the slides