Assignment 5

Due: 19 Nov 2009

Educational objectives:

- ***Primary objectives:*** Implement a simple self-restructuring binary search tree, a Queue class, and an iterator for the tree, which uses the Queue class.

- ***Secondary objectives:*** Implement and use templates, develop test cases to test the correctness of your software.

Statement of work: (i) Implement a generic self-restructuring binary search tree class, using a self-restructuring scheme specified below, (ii) implement a generic *Queue* by adapting the `Vector` class you implemented in assignment 2, and (iii) implement an iterator that uses your Queue class, as described below.

Deliverables:

- Turn in a `makefile` and all header (*.h) and cpp (*.cpp) files that are needed to build your software, as described in www.cs.fsu.edu/~asriniva/courses/DSFall09/HWinstructions.html. Turn in your development log too, which should be a plain ASCII text file called `LOG.txt` in your project directory.

Requirements:

- Create a subdirectory called `proj5`.
- You will need to have a `makefile` in this directory. In addition, all the header and cpp files needed to build your software must be present here, as well as the `LOG.txt` file.
- You should create the following additional files.
    - *Queue.h:* This should implement a generic `Queue` class with at least the following features: (i) `void push(T &)`, (ii) `void pop()`, (iii) `T &front()`, and (iv) `bool empty()`.
    - *CBST.h:* This should implement a generic `CBST` class. This class is a self-restructuring binary search tree. It restructures by `rotating` a node `N` with its parent if `N` is found by the `search` operation, and has been found at least 5 times (including the current one) by this operation. This rotation moves `N` one step closer to the root. If `N` is already the root, then it is not moved. In this type of self-organization, we are assuming that if a node is found just a few times, then it may not really be an important node; however, if it is found several times, then its importance should be related to the number of times that it has been searched for.

      You should implement at least the following features for the CBST class: (i) `void push(T &)`, (ii) `bool search(T &)`, (iii) `void PrintInorder()`, (iv) `void remove(T &)`, (v) a default constructor without arguments, and (vi) a destructor. The `push` function inserts a node into the tree. The `search` function returns `true` if and only if its argument is present in the tree, and also causes a restructuring operation to be performed. The `PrintInorder` function prints the values stored in each node, using an inorder traversal. The `void remove(T &val)` function

removes the first instance of `val` that is found in the tree. A few other features, related to the iterator, are mentioned below.

This header file should also implement a Breadth First Search iterator `BFSIterator` for the tree. This iterator will contain a `Queue` class and will return nodes in the order in which a breadth search search (level-order) traversal will visit nodes. The iterator will be valid after initialization, as long as the tree is not changed through push, remove, or search. The following operators should be defined for the iterator, with the usual meaning: (i) pre-increment operator++, (ii) dereference operator*, and (iii) comparison operators == and `!=`. The CBST class should typedef the above class to the usual name for STL iterators. It should also provide suitable `begin` and `end` member functions.

- *test.cpp:* This is a code that you write to check whether your tree works correctly. Note that we will not provide a sample executable for this assignment. You should create suitable test cases and test your code thoroughly.

- *test.txt:* This is an ASCII text file describing how you tested your code.

- *analysis.txt:* This is an ASCII text file giving the asymptotic time complexity of using your iterator to traverse a tree with `n` nodes. Justify your results.

Note:

1. We will test your `CBST` class with code that is different from your `test.cpp`. So it is important for this class to be generic and exactly as specified.
2. You will be graded for the thoroughness of your testing, and your description of it.

Last modified: 30 Oct 2009