University of Wollongong

School of Computer Science and Software Engineering

CSCI124 Applied Programming Spring 2012

Assignment 5 (Due: Week 13, Friday 26 October) 6 marks

Aim:

Programming with, pointers, dynamic memory and container classes.

On completion you should know how to:

- use classes, pointers and dynamic memory within programs,
- implement data structures for sorting and accessing data quickly,
- use appropriate data structures for data storage.

Requirements:

For this assignment you are to complete definitions of the Linked List class in **list.cpp** and the Binary Tree class in **btree.cpp**. You are to also write the incomplete function definitions in **ass5.cpp** that can enable the program to read a text file and extract statistical information on words in the file. The completed program should display a menu that provides the user with the following options:

(t) Tests the Linked List and Binary Tree classes by inserting and removing ints and words to and from instances of these objects.

(r) Asks the user for a filename from which words are extracted and inserted into a binary search tree. The tree is to also record how many times each word occurred in the file and all the locations each word was found (eg if a word was read from the 3rd, 15th and 23rd word positions in the file, then it occurred in location 3, 15 and 23). After all words are read into the tree a message should be displayed indicating how many words were read from the file.

(d) Iterates through the tree and displays on the screen each word, its frequency and all the locations where each word occurred.

An example screen dump of the completed program is available in the file **screen.txt**. This output was produced by running the program with the file **testdata.txt** as the input text file. A completed menu command interface is available in **main.cpp** and incomplete implementations of the linked list and binary tree classes are available in **list.cpp** and **btree.cpp**. The header file **ass5.h** contains function prototypes of the menu functions. The menu functions are to be implemented in **ass5.cpp** which has menu option **(t)est** already implemented for you. You should complete this assignment by performing the following steps:

Step1. Complete the class definitions of the Linked List class in **list.cpp**. The linked list class requires some functions to be implemented (see lecture notes). The datatype stored in the list and tree are defined in list.h and btree.h. You do not have to change these. When you have completed the implementation of the Linked List class test your code with the **(t)est** menu option provided in **main.cpp** and **ass5.cpp**.

Step2. Once you have your linked list working properely, complete the binary tree functions in **btree.cpp**, uncomment the rest of the TestContainers() function and test the word tree. The correct output you should expect from entering this option can be seen in the file **screen.txt**. Before going further carefully read the code comprising the TestContainers() function (and #included files) so you understand how the provided container classes can be utilised.

Step 3. Requires you to get menu option **(r)ead** working which loads a global WordTree from a text file. For this step you will also use the global instance of the BinaryTree class within **ass5.cpp**. If you are unsure how to do this take another look at the first part of the "Testing WordTree" code in TestContainers(). It can be done similarly. Make sure you remove any punctuation marks from the words read from the file and make the words lower case.

Step4. Implement the `DisplayWordStats()` function. To do this you will have to write code that iterates through the tree by using the BinaryTree's Iterator. To iterate though the tree and access each node, use the `SetIterator()`, `More()` and `Next()` member functions of the BinaryTree and LinkedList class as shown in the provided pseudo code and the `TestContainers()` function and in TestContainers(). Make sure the output from `DisplayWordStats()` complies with the format shown in screen.txt.

Note: Make sure your output complies with the format described in screen.txt and your program works with the input data supplied prior to submitting it.

Try to adhere to the implementation procedure described above. Additional information and assistance will be provided in the Labs and Lectures. Make sure your code continues to work after each step before proceeding onto the next step. (It is a good idea to always save the previous step in case you fail to complete the next step). If you are unsure of exactly what you are to do in each step do not hesitate to ask your tutor or lecturer. (The onus is on you to resolve anything you find unclear or ambiguous). Marks will be awarded according to how many of the above steps you complete (or part there of) and for the correctness of your code. Marks will be reduced for inappropriate design and code that is poorly written or difficult to understand. For further information see "Coding Guidelines.txt"

Submit:

You have to submit the following files on UNIX: **main.cpp, ass5.h, ass5.cpp, btree.h, list.h, btree.cpp and list.cpp.** Submit your files using the submit facility on UNIX ie:

$ submit -u *login* **-c CSC124 -a 5** *files.*

where: *login* **is your UNIX login ID and**
files **is the names of ALL of the above files**

The assignment deadline will be strictly enforced. An extension of time for the assignment submission may be granted in certain circumstances. Any request for an extension of the submission deadline must be made to the Subject Coordinator before the submission deadline. Supporting documentation must accompany the request for any extension. Late assignment submissions without granted extension will be marked but the mark awarded will be reduced by 1 mark for each day late. Assignments will not be accepted if more than 5 days late.