University of Wollongong

School of Computer Science and Software Engineering

CSCI124/MCS9124 Applied Programming Spring 2014

Assignment 5 (6 marks)

Due 11:59pm Sunday November 2, 2014

Background:

The assignment examines a student's knowledge of linked lists and tree's.

When a programmer designs a program, they rely on the compiler to report such errors as the mistyping of variable names. A useful tool for the programmer is a program to display all the variable names used in their program file, displaying the line numbers on which the identifier appears.

Step One (3 marks)

Write a class called `program` which opens a specified file and, upon request reads the next line of the file, breaking the line up into tokens.

A C++ program for filtering out the comments, strings and pre-preocessor directives is provided as a base. The source code can be found in the file `base.cpp`. Modifications are required so that the collecting of symbols can be limited to each line of input.

A token is defined as any valid C++ identifier. There should be a function which returns all tokens on the next line of input. These tokens should be stored in a linked list for return by the function. The function should also return a integer indicating either there are no more tokens to be read or the line number which the returned list represents. It is best to pass an empty linkedlist to this function by reference.

Use a character array to store each individual token in the list. Test the class before proceeding. Compare the output using your class to the base code's output – it does work! Place the implementation and class definition in the files `program.cpp` and `program.h` respectively. You will also need to define a linked list for `program.cpp`. You should place the code for this in `program-list.cpp` and `program-list.h`. This linked list need not be a class, but if you wish to make it a class then you can. You can use the code as demonstrated in class if need be.

Step Two (3 Marks)

The final program will store these tokens (symbols) returned from the `program` class, in a Binary Tree in the following form. For each token (the tree's key), maintain a list of line numbers on which each token occurs.

When the file is exhausted, the tree can be printed by iterating through printing out the tokens and their associated line numbers.

No line number should appear twice in the output for a token. As mentioned previously the final program will store these tokens (identifiers) in a binary tree. The identifiers are the key for searching/ordering data in the tree. In addition to this each node in the tree has a linked list which stores the line numbers the token has appeared on. You can use the code for binary trees and linked lists as discussed in class. (To do all this you will have to think of a suitable data structure for a node).

Basically the algorithm to be implemented is:

```
BEGIN main loop
    create an instance of program class
    associate program class with a file
    WHILE there are tokens to be extracted
        get a lines worth by calling the appropriate
        function from program
        FOR each token
            IF it is in the binary tree
                modify the structure i.e. add the
                line number.
            ELSE
                insert it into the tree i.e. add
                the line number and token
            ENDIF
        ENDIF
    ENDWHILE
    Traverse tree printing out tokens and line
        numbers.
END
```

Use an inorder traversal to print out the tree. Your program should accept only one piece of input - the name of the file to open for processing. An iterator has been provided in the binary tree to help you with this.

Place your code to do all this in the file `main.cpp`. Place your code for the binarytree and lined list for this section in `binarytree.cpp`, `binarytree.h`, `linkedlist.cpp` and `linkedlist.h`.

Part 3 (Bonus Part – 1 mark)

The following 62 reserved words of C++ should not be printed out. Thus they should be placed in a suitable structure and each token checked against the words prior to being output (or even put into the tree).

```
asm                 auto           bool              break
casecatch           char           class             const
const_cast          continue       default           delete
do                  double         dynamic_cast      else
enum                explicit       extern            false
float               for            friend            goto
if                  inline         int               long
mutable             namespace      new               operator
private             protected      public            register
reinterpret_cast    return         short             signed
sizeof              static         static_cast       struct
switch              template       this              throw
true                try            typedef           typeid
typename            union          unsigned          using
virtual             void           volatile          wchar_t
while
```

Submission Procedure

To submit the assignment enter the following command.

```
submit –u USERNAME –c CSCI124 –a ass5 program.cpp program.h program-list.cpp program-list.h main.cpp linkedlist.cpp linkedlist.h binarytree.cpp binarytree.h
```

Please note assignments that are late, will attract a 1 mark penalty each day late. Assignments will receive a zero mark if they are submitted more then 3 days late.