University of Wollongong

School of Computer Science and Software Engineering, Faculty of Informatics

CSCI124/MCS9124 Applied Programming Autumn 2015

Lab 7 (1 mark)

Due Thurs April 30th 11:59pm.

Background:

The aim of this lab is to get some experience with pointers and dynamic memory.

Task:

Question 1

The operator new allows objects of any type to be allocated on the heap. Consider the following structure:

```
struct student
{
     char name[128];
     int age;
     int courseno;
};
```

a) Using the new operator write the appropriate statement to create a dynamic array of `n` student instances. Assume `n` is an integer already defined.

b) Dynamic memory means more responsibility for the programmer. Write the appropriate statement to delete the dynamically allocated array you created in part a) of this question.

Place your solution in `solution.txt`.

Question 2

Refer to the source code `main.cpp` and the header `linkedList.h`.

The program uses a linked list similar to that discussed in lectures. Read the `main` file carefully. The main program creates a single linked list referred to as `list`. The linked list is essentially a collection of `word` nodes. Each node contains a string and an integer. The string represents a word and the integer represents the number of times that word has been seen. The main function opens a text file (one provided) and reads in each word of the text file into the linkedlist. After populating the list, the contents is printed out along with

the number of times each word has been seen.

The linked list has several methods.

`void initialise(word*& head)`
Initialize the linked list `head` so that it is pointing to NULL.

`void addnode(char data[], word*& head)`
Add a node to the linked list. If the word already exists in the linked list referred to by `head` – increment the count variable in the node in the list. If the word is new add the word to the end of the list and set count to zero. The function is passed a character array representing the word to insert.
`void printnodes(word*& head)`
Print the contents of each node in the linked list to standard output from beginning to end. This means it prints out each word along with the number of times it has been seen.

`void freelist(word*& head)`
Delete each node of the linked list using the delete operator and reset the linked list so that it is pointing to NULL.

Your challenge is to implement the above functions. A set of function stubs has been provided for you.
Place your implementations in `linkedlist.cpp`.

Submission

You are to submit the following files:

```
solution.txt
linkedlist.cpp
```

You should submit all the above files using the submit program:

```
$ submit –c csci124 –a lab7 –u <username> solution.txt
linkedlist.cpp
```