COSC 220 - Computer Science II
Lab 3

Dr. Joe Anderson

Due: 17 September 2019

1 Objectives

1. Practice using C++ classes
2. Develop familiarity with linked lists, and dynamic memory

2 Description

You should read the entire set of instructions before beginning the lab.

You will expand upon the `PayRoll` class you developed in the last lab. One possible application of this class is that, if you were the manager of a company, you would benefit from software which stores and processes your employees and their data, helping you to manage their paychecks each week. Instead of an array – which you would have seen can be very brittle to use – you will store several `PayRoll` objects in a linked list, then use that list to generate the paychecks for each employee stored in the list.

3 Tasks

1. Design a PayRoll class that has data members for an employees name, hourly pay rate, number of hours worked, and total pay for the week
   (a) **You may modify the PayRoll class you designed last lab!**
   (b) The PayRoll class should implement the following prototype:

```
class PayRoll {
  private:
    string name;
    double rate;
    double hours;

  public:
    PayRoll(); // ctor
    PayRoll(string, double, double); // non-default ctor
    double getTotal();
    double getRate();
    double getHours();
    void setRate(double);
    void setName(string);
    void setHours(double);
};
```

2. Implement a linked list whose nodes store PayRoll objects.
   (a) The PayRoll objects should be sorted by pay rate within the list.
   (b) The list should be called "PayRollList" and implement the following class definition:

```
class PayRollList{
  private:
    struct ListNode {
      PayRoll p;
      ListNode* next;
    };

    ListNode* head;

  public:
    PayRollList(); // ctor
    ~PayRollList(); // destructor, do this carefully!

    // Takes name, rate, and hours worked as
    // parameters for a new ListNode.
    // The new node may be added to any part
    // of your linked list
    void insert(string, double, double);

    // Override, accept a PayRoll object directly.
    // To make life easy, this should either call or
    // be called from insert(string, double, double)
    // rather than repeat the logic!
    void insert(PayRoll);

    // Prints each employee name and total pay to the terminal
    // or, optionally, to a text file (loosely simulating the
    // user pressing the "print" button in a program)
    void printPayChecks();
};
```

   (c) Note that this class depends on the `PayRoll` class, so will need to `#include` the appropriate header file.
3. Construct your `main` function to test each of the methods above with several hard-coded examples.
4. Next, modify your main program to read a list of employee names, rates, and hours **from a file**. You may design and use your own convention for the file format, but be sure to include your sample data with your submission.
5. The main function should then print the paychecks for each employee by calling the `printPayChecks` method on the list object.
6. The header files for your classes should use proper include guards.

7. Check your program for memory leaks with `valgrind` and make sure your constructors and destructors appropriately handle your dynamic memory.

8. Include a `README` file to document your program, provide instructions to compile and run it, explain how it works, and provide a reference for the data format of your input file.

9. Your code should be organized between the files: `main.cpp`, `payroll.h`, `payroll.cpp`, `payrolllist.h`, and `payrolllist.cpp`.

4 Submission

Upload your project files to the course canvas system in a single `.zip` file.

Turn in (stapled) printouts of your project files, properly commented and formatted with your name, course number, and complete description of the code.

Also turn in printouts reflecting several different runs of your program (you can copy/past from the terminal output window). Be sure to test different situations, show how the program handles erroneous input and different edge cases.

5 Bonus

(10 pts) Use the guides linked from the course webpage to write a `Makefile` which automatically compiles your code by first compiling into three object files (`main.o`, `payroll.o`, and `payrolllist.o`) and then linking into a single executable.