COSC 220 - Computer Science II
Lab 5

Dr. Joe Anderson

1 October 2019

1 Objectives

1. Practice linked lists, dynamic memory
2. Practice implementing stacks
3. Practice using the `make` utility and creating a `Makefile`

2 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) This may already be complete from Lab 3, but you should double-check your implementation.
    (b) The PayRoll class should implement the following prototype:

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

  public:
    // Default Constructor
    PayRoll();

    // Constructor with passed values
    PayRoll(string, double, double);

    // Prints the check information for the employee
    void printCheck();

    // Standard mutators
    double getTotal();
    double getRate();
    double getHours();
    void setRate(double);
    void setName(string);
    void setHours(double);
```

```
        };
```

2. Implement a linked list-based stack whose nodes contain PayRoll objects.
   (a) The stack will use a linked list to manage the underlying data.
   (b) Recall the two needed operations for a stack (first in, last out):
      i. push: add an item to the stack.
      ii. pop: remove and return the *most*-recently pushed item.
   (c) The class should be called "PayRollStack" and implement the following class definition:

```
class PayRollStack{
  private:
    struct StackNode {
      PayRoll p;
      StackNode* next;
    };

    StackNode* head;

  public:
    PayRollStack();

    // Rule of three!
    ~PayRollStack();
    PayRollStack(const PayRollStack&);
    PayRollStack operator=(const PayRollStack&);

    // Adds a PayRoll object into the stack
    void push(PayRoll p);

    // Returns the payroll object that was most recently pushed
    PayRoll pop();

    // Alternative, but common, method:
    // instead of using a return, take the destination container
    // as a parameter to the pop function
    void pop(PayRoll&);

    // Returns the number of elements on the stack
    int size();
};
```

3. The main program should instantiate a stack and insert seven employees using your file input from last lab.
4. Then be sure to test and demonstrate correct behavior for each of the above implemented methods.
5. When implementing the overloaded assignment operator, be sure to prevent self-assignment, and free any memory that is no longer needed after the copy.

6. To test the “FILO” behavior, display each employee’s name and total wages for that week by popping all PayRoll objects from a stack until it is empty. Check your output: are the employees printed in reverse order of how they were added?

7. Write a `Makefile` to compile your code.

   (a) It may be simple, e.g. one target:

   ```
   all:
     g++ -std=c++11 -o lab3 payrollstack.cpp payroll.cpp main.cpp
   ```

   (b) It may be more complicated, with variables, more targets, e.g. if you have three .cpp files (main.cpp, PayRoll.cpp, PayRollStack.cpp):

   ```
   all: lab

   lab: main.o payrollstack.o payroll.o
     g++ -std=c++11 -o lab main.o payrollstack.o payroll.o

   main.o: main.cpp
     g++ -std=c++11 -c main.cpp

   payrollstack.o: payrollstack.cpp
     g++ -std=c++11 -c payrollstack.cpp

   payroll.o: payroll.cpp
     g++ -std=c++11 -c payroll.cpp
   ```

3 Submission

Upload your project files to the course canvas system in a single zipped folder: To zip on Linux:

1. To zip a single folder into a single archive called “myZip.zip”:

   ```
   zip -r myZip.zip folderName
   ```

2. To zip multiple files into a zip file called “myZip.zip”:

   ```
   zip myZip.zip file1.cpp file2.h file3 file4.cpp
   ```

Turn in (stapled) printouts of your source code, 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.