CSCI 140 PA 8 Submission

Due Date: 4/25/23

Name: Ali Mortada

Exercise 1 – **zyBook 8.14 LAB: Library book sorting** -- check if completely done in zyBook __X__ ; otherwise, discuss issues below
Include a screenshot of current status/score

LAB ACTIVITY | 8.14.1: LAB: Library book sorting | 10 / 10

Current file: main.cpp | Load default template...

```
    currNode = new BookNode(bookTitle, bookAuthor, bookISBN);
    // TODO: Call LL_Library's InsertSorted() method to insert currNode and return
    //       the number of operations performed
    linkedListOperations = linkedListLibrary.LinkedListLibrary::InsertSorted(currNode);


    // Insert into VectorList
    tempBook  = Book(bookTitle, bookAuthor, bookISBN);
    // TODO: Call VectorLibrary's InsertSorted() method to insert tempBook and return
    //       the number of operations performed
    vectorOperations = vectorLibrary.VectorLibrary::InsertSorted(tempBook);

    // TODO: Print number of operations for linked list
    cout << "Number of linked list book copy operations: " << linkedListOperations << endl;

    // TODO: Print number of operations for vector
    cout << "Number of vector book copy operations: " << vectorOperations << endl;

    return 0;
}
```

Exercise 2 – **zyBook 8.17 LAB: Playlist** -- check if completely done in zyBook __X__ ; otherwise, discuss issues below
Include a screenshot of current status/score

LAB ACTIVITY | 8.17.1: LAB: Playlist (output linked list) | 10 / 10

Current file: main.cpp | Load default template...

```
#include <iostream>
#include "SongNode.h"

// TODO: Write PrintPlaylist() function
void PrintPlaylist(SongNode *headNode) {
   SongNode *currNode;

   currNode = headNode->GetNext();
   while (currNode != nullptr) {
      currNode->PrintSongInfo();
      currNode = currNode->GetNext();
      if (currNode != nullptr) {
         cout << endl;
      }
   }
}

```

Exercise 3 – **Dynamic Array** -- check if completely done __X__ ; otherwise, discuss issues below

Source code below:

```
/*
    Program: Dynamic Array
    Author: Ali Mortada
    Class: CSCI 140
    Date: 4/27/23
    Description: Prompts user for monthly sales and stores them in a dynamically
allocated
    array. Calls a function to fill in the array with user input, then computes the
total
    and average monthly sales and displays them with two digits of precision. Returns
allocated
    memory when done.
    I certify that the code below is my own work.
    Exception(s): N/A
*/

#include <iostream>
#include <iomanip>
using namespace std;

void inputMonthlySales(double *, int);
double computeTotalSales(double *, int);
double computeAvgSales(double *, int);

int main() {

    int numMonthlySales;

    cout << "Author: Ali Mortada" << endl << endl;

    // Prompt user for number of monthly sales
    cout << "Please enter the number of monthly sales --> ";
    cin >> numMonthlySales;

    // Fill data into dynamic array
    double *dynamicArray = new double[numMonthlySales];
    inputMonthlySales(dynamicArray, numMonthlySales);

    // Output total and average monthly sales
```

```
    cout << "Total sales: $" << fixed << setprecision(2) <<
computeTotalSales(dynamicArray, numMonthlySales) << endl;
    cout << "Average monthly sales: $" << fixed << setprecision(2) <<
computeAvgSales(dynamicArray, numMonthlySales) << endl;

    // Delete array
    cout << "Returning dynamic memory..." << endl;
    delete[] dynamicArray;
    cout << "Done." << endl;

    return 0;
}

void inputMonthlySales(double *dynamicArray, int numMonthlySales) {
    // Keep prompting user to input data into array
    for (int i = 0; i < numMonthlySales; i++) {
        cout << "Please input the sales for month " << i + 1 << " --> ";
        cin >> dynamicArray[i];
    }
    cout << endl;
}

double computeTotalSales(double *dynamicArray, int numMonthlySales) {
    double totalSales = 0;
    // Iterate through array and add up all the sales
    for (int i = 0; i < numMonthlySales; i++) {
        totalSales += dynamicArray[i];
    }

    return totalSales;
}

double computeAvgSales(double *dynamicArray, int numMonthlySales) {
    double totalSales = 0;
    double averageSales = 0;
    // Add up all sales
    for (int i = 0; i < numMonthlySales; i++) {
        totalSales += dynamicArray[i];
    }

    // Divide by total to get average
    averageSales = totalSales / numMonthlySales;

    return averageSales;
}
```

Input/output below:

```
Author: Ali Mortada

Please enter the number of monthly sales --> 4
Please input the sales for month 1 --> 1290.89
Please input the sales for month 2 --> 905.95
Please input the sales for month 3 --> 1567.98
Please input the sales for month 4 --> 994.83

Total sales: $4759.65
Average monthly sales: $1189.91
Returning dynamic memory...
Done.
```

Exercise 4 – **MyInteger Class** -- check if completely done __X__ ; otherwise, discuss issues below

Source code below:

Driver program:

```
// A driver to test MyInteger class
// Created and updated by T. Vo for CSCI 140 Spring 2023.

// Modified by: Ali Mortada

#include <iostream>
#include "MyInteger.h"

using namespace std;

int main()
{
    MyInteger i1;       // 0
    MyInteger i2(5);    // 5 (copy constructor)
    MyInteger i3 = i2;  // 5 (copy constructor)
    MyInteger* pMyInt;  // a pointer

    cout << "i1: " << i1 << endl;               // 0
    cout << "i2: " << i2.getValue() << endl;    // 5
    cout << "i3: " << i3 << endl;               // 5

    i1.setValue(-4);
    i3 = i1 + i2;
    cout << "i3: " << i3 << endl;              // 1

    cout << "(i2 - i1) % 2: " << (i2 - i1) % 2 << endl;    // 1
    cout << "i2 * i1 / 2: " << i2 * i1 / 2 << endl;    // -10

    cout << "Enter an integer: ";
    cin >> i1;                                 // input 123

    cout << i1 << " == " << i2 << ": ";
    if (i1 == i2)                              // false
        cout << "true" << endl;
    else
        cout << "false" << endl;

    i2 = i1;                                   // copy assignment operator
```

```
    cout << i1 << " != " << i2 << ": ";
    if (i1 != i2)                               // false
        cout << "true" << endl;
    else
        cout << "false" << endl;

    i2.setValue(25);
    cout << "i1: " << i1 << endl;               // 123
    cout << "i2: " << i2 << endl;               // 25

    pMyInt = new MyInteger(i2);                 // 25 (copy constructor)
    cout << "*pMyInt: " << *pMyInt << endl;     // 25
    *pMyInt = i1;                               // 123 (copy assignment
operator)
    cout << "pMyInt->getValue(): " << pMyInt->getValue() << endl;   // 123
    delete pMyInt;                              // return allocated memory

    // feel free to add more test cases below

    cout << "End of test cases." << endl;
    return 0;
}
```

Class declaration:

```
// Created and updated by T. Vo for CSCI 140 Spring 2023.
// Do not modify this file.
#ifndef MYINTEGER_H
#define MYINTEGER_H

#include <iostream>

using namespace std;

class MyInteger
{
public:
    MyInteger(int v = 0);
    MyInteger(const MyInteger &origInteger);         // Copy constructor
    void setValue(int v);
    int getValue() const;
    void operator=(const MyInteger &r);              // Copy assignment operator
    MyInteger operator+(const MyInteger &r) const;
    MyInteger operator-(const MyInteger &r) const;
    MyInteger operator*(const MyInteger& r) const;
```

```
    MyInteger operator/(const MyInteger& r) const;
    MyInteger operator%(const MyInteger& r) const;
    bool operator==(const MyInteger &r) const;
    bool operator!=(const MyInteger& r) const;

    friend ostream &operator<<(ostream &out, const MyInteger &r);
    friend istream &operator>>(istream &in, MyInteger &r);

private:
    int value;
};
#endif
```

Class definition:

```
#include <iostream>
#include "MyInteger.h"
using namespace std;

MyInteger::MyInteger(int v /*= 0*/) {
    value = v;
}

// Copy constructor
MyInteger::MyInteger(const MyInteger &origInteger) {
    value = origInteger.value;
}


void MyInteger::setValue(int v) {
    value = v;
}

int MyInteger::getValue() const {
    return value;
}

void MyInteger::operator=(const MyInteger &r) {
    value = r.value;
}

MyInteger MyInteger::operator+(const MyInteger &r) const {
    return value + r.value;
}
```

```
MyInteger MyInteger::operator-(const MyInteger &r) const {
    return value - r.value;
}

MyInteger MyInteger::operator*(const MyInteger& r) const {
    return value * r.value;
}

MyInteger MyInteger::operator/(const MyInteger& r) const {
    return value / r.value;
}

MyInteger MyInteger::operator%(const MyInteger& r) const {
    return value % r.value;
}

bool MyInteger::operator==(const MyInteger &r) const {
    if (value == r.value) {
        return true;
    }
    return false;
}

bool MyInteger::operator!=(const MyInteger& r) const {
    if (value != r.value) {
        return true;
    }
    return false;
}

ostream &operator<<(ostream &out, const MyInteger &r) {
    out << r.value;
    return out;
}

istream &operator>>(istream &in, MyInteger &r) {
    in >> r.value;
    return in;
}
```

Input/output below:

```
PS C:\Users\Ali\Documents\Mt. SAC\CSCI 140\CSCI-140\PA 8> g++ testMyInteger.cpp MyInteger.cpp -o MyInteger
PS C:\Users\Ali\Documents\Mt. SAC\CSCI 140\CSCI-140\PA 8> ./MyInteger
i1: 0
i2: 5
i3: 5
i3: 1
(i2 - i1) % 2: 1
i2 * i1 / 2: -10
Enter an integer: 123
123 == 5: false
123 != 123: false
i1: 123
i2: 25
*pMyInt: 25
pMyInt->getValue(): 123
End of test cases.
```

Answer for Question 1

One advantage of working with dynamic memory is that it allows you to free up unused memory to be used for other purposes. One challenge of working with dynamic memory is that the programmer must remember to free up memory when it is not being used, or else the program could result in a memory leak and crash.

Answer for Question 2

A friend function is a type of function that is declared outside of a class's scope but has access to its private members. They don't have to be used, but are helpful when linking classes together since a member function of one class can be declared as a friend function of another class, giving it access to the private data members of the second class. Friend functions need to be used when overloading the stream insertion and extraction operators, though.

Extra Credit – provide if applicable.

Source code:

Class declaration:

Class definition:

Input/output: