CSCI 140 PA 12 Submission

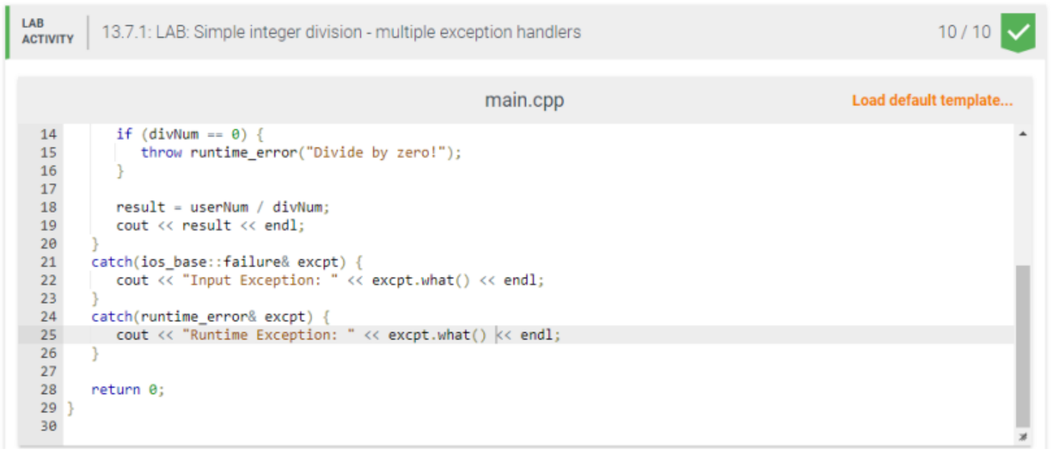
Due Date: 5/30/23

Name(s): Ali Mortada & Amber Ellis

Amber: Exercises 1, 2, and 3

Ali: Exercise 4, Questions 1 and 2, and EC

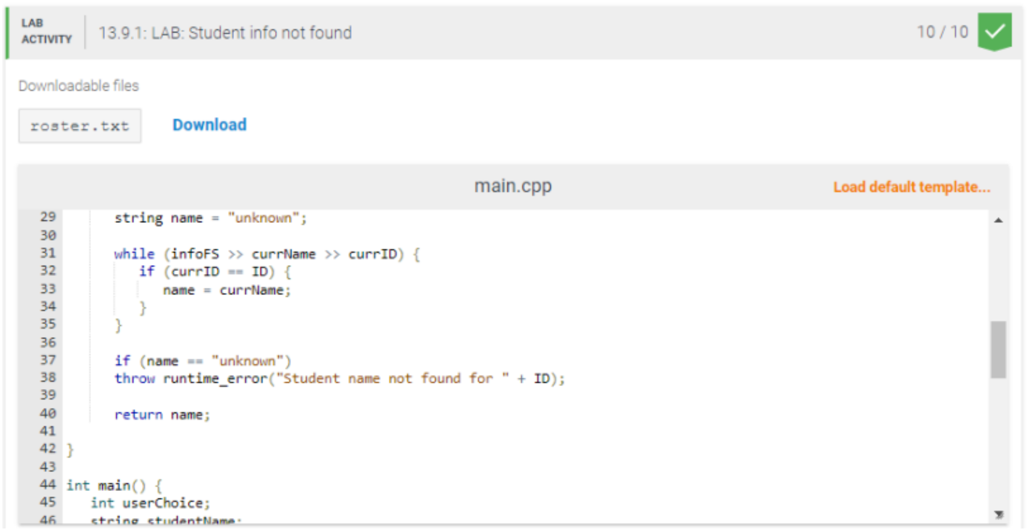
Exercise 1 – **zyBook 13.7 LAB: Simple integer division – multiple exception handlers** -- check if completely done in zyBook __X__ ; otherwise, discuss issues below


LAB ACTIVITY | 13.7.1: LAB: Simple integer division - multiple exception handlers 10 / 10

main.cpp

```
      if (divNum == 0) {
         throw runtime_error("Divide by zero!");
      }

      result = userNum / divNum;
      cout << result << endl;
   }
   catch(ios_base::failure& excpt) {
      cout << "Input Exception: " << excpt.what() << endl;
   }
   catch(runtime_error& excpt) {
      cout << "Runtime Exception: " << excpt.what() << endl;
   }

   return 0;
}
```

Exercise 2 – **zyBook 13.9 LAB: Student info not found** -- check if completely done in zyBook __X__ ; otherwise, discuss issues below


LAB ACTIVITY | 13.9.1: LAB: Student info not found 10 / 10

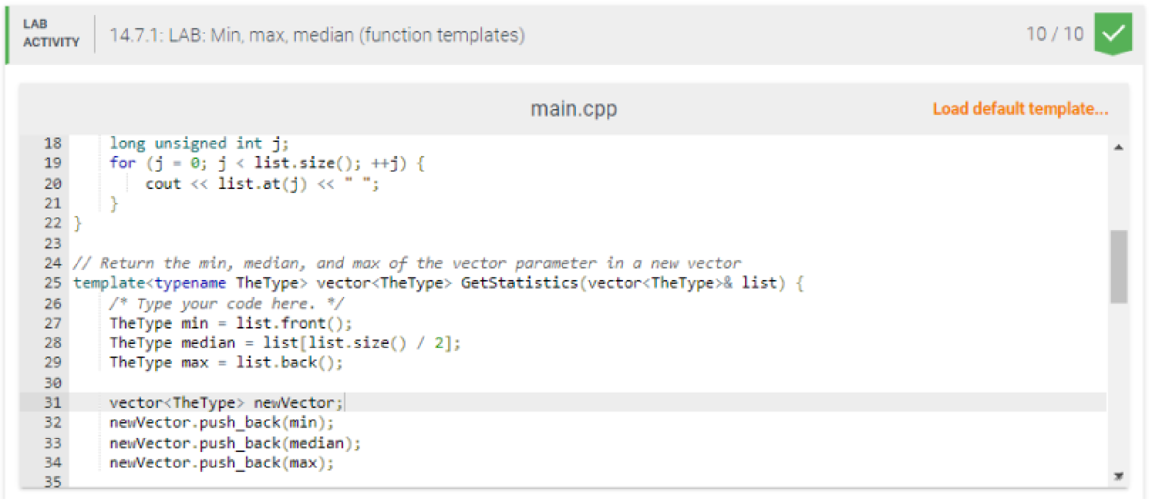
Downloadable files

roster.txt Download

main.cpp

```
   string name = "unknown";

   while (infoFS >> currName >> currID) {
      if (currID == ID) {
         name = currName;
      }
   }

   if (name == "unknown")
   throw runtime_error("Student name not found for " + ID);

   return name;

}

int main() {
   int userChoice;
   string studentName;
```

Exercise 3 – **zyBook 14.7 LAB: Min, max, median (function templates)** -- check if completely done in zyBook __X__ ; otherwise, discuss issues below



```cpp
    long unsigned int j;
    for (j = 0; j < list.size(); ++j) {
        cout << list.at(j) << " ";
    }
}

// Return the min, median, and max of the vector parameter in a new vector
template<typename TheType> vector<TheType> GetStatistics(vector<TheType>& list) {
    /* Type your code here. */
    TheType min = list.front();
    TheType median = list[list.size() / 2];
    TheType max = list.back();

    vector<TheType> newVector;
    newVector.push_back(min);
    newVector.push_back(median);
    newVector.push_back(max);

```

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

Source code below:

Header file:

```cpp
/*
    Program: Class Templates - MyNum
    Author: Ali Mortada
    Class: CSCI 140
    Date: 5/29/23
    Description: Copy of MyInteger class from PA 8, but uses a class template to allow
use for other data types such as double, etc.
    I certify that the code below is my own work.
    Exception(s): N/A
*/

#ifndef MYNUM_H
#define MYNUM_H

#include <iostream>

using namespace std;

template<typename NumType>
```

```cpp
class MyNum
{
public:
    MyNum(NumType v = 0) {
        value = v;
    }

    MyNum(const MyNum &origNumber) {            // Copy constructor
        value = origNumber.value;
    }

    void setValue(NumType v) {
        value = v;
    }

    int getValue() const {
        return value;
    }

    void operator=(const MyNum &r) {            // Copy assignment operator
        value = r.value;
    }

    MyNum operator+(const MyNum &r) const {
        return value + r.value;
    }

    MyNum operator-(const MyNum &r) const {
        return value - r.value;
    }

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

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

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

    bool operator==(const MyNum &r) const {
        if (value == r.value)
```

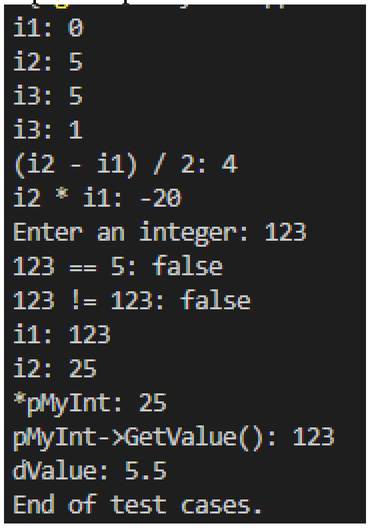
```
            return true;
        return false;
    }

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

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

    friend istream &operator>>(istream &in, MyNum &r) {
        in >> r.value;
        return in;
    }

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

Driver program:

```
// A driver to test MyNum class
// Created and updated by T. Vo for CSCI 140 Fall 2022.

// Modified by: Ali Mortada

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

using namespace std;

int main()
{
    MyNum<int> i1;      // 0
    MyNum<int> i2(5);   // 5 (copy constructor)
    MyNum<int> i3 = i2; // 5 (copy constructor)
    MyNum<int>* 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;    // 4
    cout << "i2 * i1: " << i2 * i1 << endl;    // -20

    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 MyNum<int>(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

    // try double
    MyNum<double> dValue(5.5);
    cout << "dValue: " << dValue << endl;      // 5.5

    // feel free to add more test cases below

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

```
}
```

Input/output below:



```
i1: 0
i2: 5
i3: 5
i3: 1
(i2 - i1) / 2: 4
i2 * i1: -20
Enter an integer: 123
123 == 5: false
123 != 123: false
i1: 123
i2: 25
*pMyInt: 25
pMyInt->GetValue(): 123
dValue: 5.5
End of test cases.
```

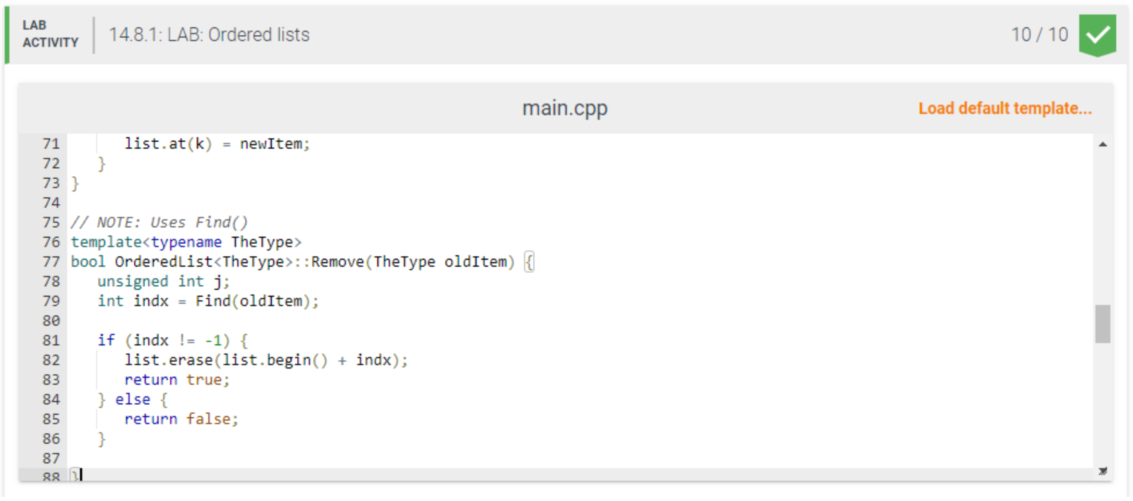
Answer for Question 1

Exceptions should be handled in a program in order to make sure that the program works as intended. For example, if a program prompts a user for input, the user may input unexpected or bad data. If the program cannot handle exceptions, it could cause unanticipated errors or not work properly.

Answer for Question 2

Function templates should be chosen over function overloading when the programmer wants to write multiple functions that perform the same operation but differ with data types. This is because with a function template, only one function needs to be written to handle multiple data types, but with function overloading, the same function needs to be rewritten multiple times for each data type. Thus, to avoid redundancy, function templates should be used in this case.

Extra Credit – **zyBook 14.8 LAB: Ordered lists**



```
        list.at(k) = newItem;
    }
}

// NOTE: Uses Find()
template<typename TheType>
bool OrderedList<TheType>::Remove(TheType oldItem) {
    unsigned int j;
    int indx = Find(oldItem);

    if (indx != -1) {
        list.erase(list.begin() + indx);
        return true;
    } else {
        return false;
    }

}
```