CSCI 140 PA #1 Submission

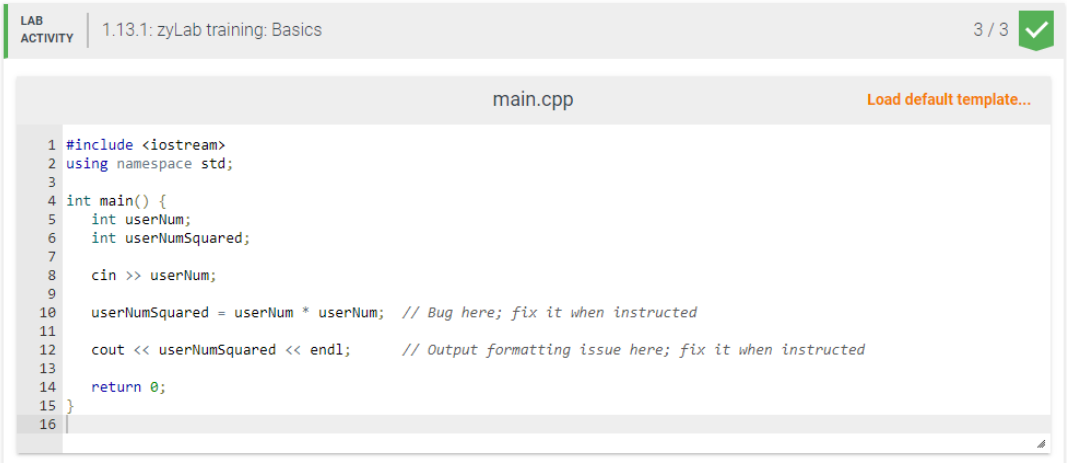
Due Date: 2/28/23

Name: Ali Mortada

Exercise 1 – provide zyBook lab such as **zyBook 1.13 zyLab training: Basics** as applicable

-- check if completely done in zyBook __X__ ; otherwise, discuss issues below

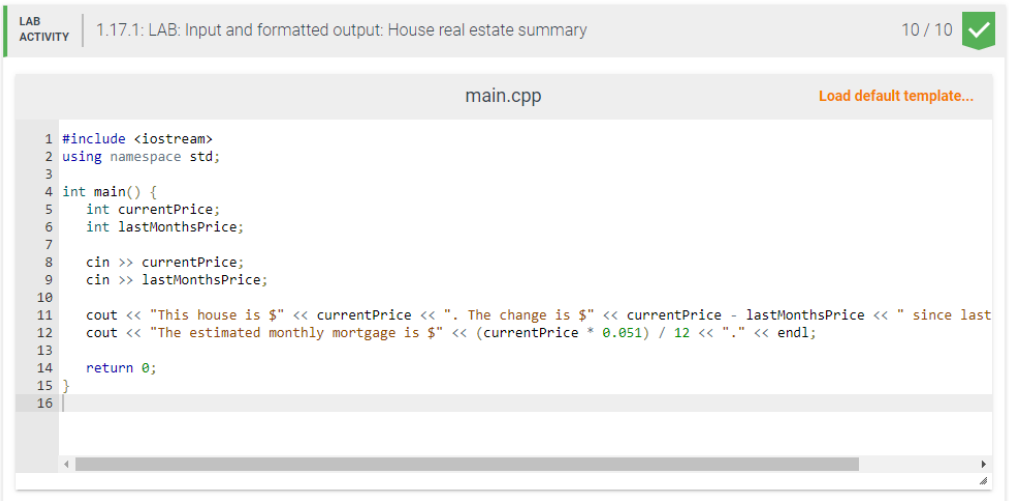
Include a screenshot of current status/score



Exercise 2 – **zyBook 1.17 LAB: Input and formatted output: House real estate summary**

-- check if completely done in zyBook _X_ ; otherwise, discuss issues below

Include a screenshot of current status/score



Exercise 3 – **zyBook 1.21 LAB*: Program: ASCII art**

-- check if completely done in zyBook _X_ ; otherwise, discuss issues below

Include a screenshot of current status/score

LAB ACTIVITY | 1.21.1: LAB*: Program: ASCII art 5 / 5

main.cpp Load default template...

```cpp
#include <iostream>
using namespace std;

int main() {

   // Draw tree
   cout << "   *"    << endl;
   cout << "  ***"   << endl;
   cout << " *****"  << endl;
   cout << "*******" << endl;
   cout << "  ***"   << endl;

   // Create two blank lines
   cout << endl; cout << endl;

   // Draw cat
   cout << "/\\   /\\" << endl;
   cout << "  o o" << endl;
   cout << " =   =" << endl;
   cout << "  ---" << endl;

   return 0;
}
```

Exercise 4 – **zyBook 2.27 LAB: Driving costs**

-- check if completely done in zyBook _X_ ; otherwise, discuss issues below

Include a screenshot of current status/score

LAB ACTIVITY | 2.27.1: LAB: Driving costs 10 / 10

main.cpp Load default template...

```cpp
#include <iostream>
#include <iomanip>      //For setprecision
using namespace std;

int main() {

   // Declare variables and prompt user for input
   double milesPerGallon, dollarsPerGallon, dollarsPerMile;
   cin >> milesPerGallon >> dollarsPerGallon;

   // Calculate dollars per mile
   dollarsPerMile = dollarsPerGallon / milesPerGallon;

   cout << fixed << setprecision(2);   // All later cout's will print floating-point values to exactly 2 decimal plac
                                       // Ex: 3.60

   // Output gas cost for 20, 75, and 500 miles
   cout << dollarsPerMile * 20 << " " << dollarsPerMile * 75 << " " << dollarsPerMile * 500 << endl;

   return 0;
}
```

Exercise 5 – **zyBook 2.29 LAB: Using math functions**

-- check if completely done in zyBook _X_ ; otherwise, discuss issues below

Include a screenshot of current status/score

LAB ACTIVITY | 2.29.1: LAB: Using math functions — 10 / 10

main.cpp — Load default template...

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main() {
   // Declare variables and prompt user for input
   double x, y, z;
   cin >> x >> y >> z;

   // Calculate and show outputs
   cout << pow(x, z) << " ";
   cout << pow(x, pow(y, z)) << " ";
   cout << abs(y) << " ";
   cout << sqrt(pow(x * y, z)) << endl;

   return 0;
}
```

Exercise 6 – **zyBook 2.36 LAB*: Program: Painting a wall**

-- check if completely done in zyBook _X_ ; otherwise, discuss issues below

Include a screenshot of current status/score

LAB ACTIVITY | 2.36.1: LAB*: Program: Painting a wall — 10 / 10

main.cpp — Load default template...

```cpp
#include <iostream>
#include <iomanip>              // needed for setprecision() and fixed
#include <cmath>                // needed for ceil()
using namespace std;

int main() {
   // Declare variables and prompt user for input
   double wallHeight, wallWidth, wallArea, paintCanCost, paintNeeded, cansNeeded, paintCost, salesTax, totalCost;
   cin >> wallHeight >> wallWidth >> paintCanCost;

   // Calculate wall area
   wallArea = wallHeight * wallWidth;
   // Calculate paint needed
   paintNeeded = wallArea / 350;
   // Calculate cans needed
   cansNeeded = int(ceil(paintNeeded));
   // Calculate paint cost, sales tax, and total cost
   paintCost = paintCanCost * cansNeeded;
   salesTax = .07 * paintCost;
   totalCost = paintCost + salesTax;

   // Print output
   cout << fixed << setprecision(1) << "Wall area: " << wallArea << " sq ft" << endl;
   cout << fixed << setprecision(3) << "Paint needed: " << paintNeeded << " gallons" << endl;
   cout << fixed << setprecision(0) << "Cans needed: " << cansNeeded << " can(s)" << endl;
   cout << fixed << setprecision(2) << "Paint cost: $" << paintCost << endl;
   cout << fixed << setprecision(2) << "Sales tax: $" << salesTax << endl;
   cout << fixed << setprecision(2) << "Total cost: $" << totalCost << endl;


   return 0;
}
```

Exercise 7 -- use for a non-zyBook exercise; need to submit source code and I/O
  -- check if completely done __X__ ; otherwise, discuss issues below
Pseudocode below if applicable:

-create variables totalMinutes, hours, and minutes
-prompt user for input and store value in totalMinutes
-integer divide totalMinutes by 60 to get number of hours, store it in hours
-modulus divide totalMinutes by 60 to get remaining number of minutes, store it in minutes
-output number of hours and leftover minutes

Source code below:

```
/*  Program: timeConversion
    Author: Ali Mortada
    Class: CSCI 140
    Date: 2/28/2023
    Description: Inputs number of minutes and outputs number of hours and leftover minutes.
    I certify that the code below is my own work.
    Exception(s): N/A
*/

#include <iostream>
using namespace std;

int main(){

    // Initialize variables
    int totalMinutes, hours, minutes;

    // Prompt user for input
    cout << "Enter total number of minutes: ";
    cin >> totalMinutes;

    // Calculate hours and leftover minutes
    hours = totalMinutes / 60;
    minutes = totalMinutes % 60;

    // Output number of hours and leftover minutes
    cout << totalMinutes << " equals " << hours << " hour(s) and " << minutes << " minute(s)." << endl;

    return 0;
}
```

Input/output below:

```
[alimortada@Ali-Mortadas-MacBook-Air CSCI 140 % ./timeConversion
Enter total number of minutes: 100
100 equals 1 hour(s) and 40 minute(s).
[alimortada@Ali-Mortadas-MacBook-Air CSCI 140 % ./timeConversion
Enter total number of minutes: 601
601 equals 10 hour(s) and 1 minute(s).
```

Answer for Question 1

For a software developer, source code is much more important than executable code because source code is much easier to understand and edit. Executable code is written in machine language, which is understood by the computer but very difficult to parse by humans. Thus, software developers write source code and use compilers and assemblers to translate their source code into executable code that the machine understands.

Answer for Question 2

If the program does not give any errors and gives the desired output when given a specific input, then that is enough evidence to show that the program is working correctly.

Extra Credit – provide if applicable