CSE 1321L: Programming and Problem Solving I Lab

Assignment 2 – 100 points

Solving Problems

What students will learn:
1) Problem solving
2) Concepts from assignment 1 (I/O, variables, intermediate calculations, operators)
3) Type casting

Overview: For this assignment, you're going to continue writing programs that solve problems. Again, start early, practice, and ask a lot of questions.

*Some of you asked us how the autograder works, and it's relatively simple. The autograder pushes "user" input into your program and writes out a file of your program's output. Then, it compares that against a solution file. If the output matches, you get full points for that assignment. Though there is *some* flexibility in the comparison, the autograder still requires you to match input/output exactly.*

From us to you: We recommend that you start early and contact us throughout the week since that gives us enough time to answer your questions.

Finally, remember that you need to learn everything you can, so **don't cheat**. While graduating from KSU may get you an interview, the more answers you get correct in that technical interview, the higher the salary is likely to be. You can also show off at nerd parties!

Assignment 2A:

Finding multiples We've learned that in programming, there is a special operator for getting the whole number remainder of division: The Modulo Operator! So, 12 % 9 will give us a value of 3, since 9 goes evenly into 12 once with a remainder of 3.

There are many uses for this operator which we will learn throughout the semester. In this assignment, we're going to use it to find the nearest multiple of a number based on user input. For example, if we're looking for multiples of 6, then the nearest multiple from 13 is *12*, and the nearest multiple from 22 is *18* (rounded down).

Your task is to:

a) Ask the user to enter a whole number they want to find a multiple of
b) Read that value in
c) Ask the user to enter a second number
d) Read that value in
e) Use the **Modulo Operator** to find the nearest multiple of the first number from the second number
f) Display the result to the user

The algorithm's output is as shown below, with user input in bold. Save your source code in a file called **Assignment2A** (with a file extension of **.cpp**, **.cs** or **.java**)

Sample Output #1:

```
Enter a number you want to find a multiple of: 3
Enter a second number: 10
Calculating...
The nearest multiple of 3 from 10 is 9!
```

Sample Output #2:

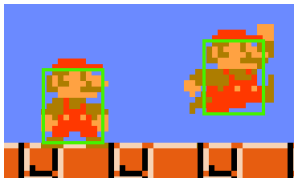
```
Enter a number you want to find a multiple of: 12
Enter a second number: 95
Calculating...
The nearest multiple of 12 from 95 is 84!
```

Sample Output #3:

```
Enter a number you want to find a multiple of: 7
Enter a second number: 33
Calculating...
The nearest multiple of 7 from 33 is 28!
```

Assignment 2B:

*Hit Boxes (Part 1)***!** In many video games, an invisible rectangle called a "hit box" surrounds the player, enemies, and parts of the environment you can interact with. This hit box is used (behind the scenes) to determine when a player collides with items and hazards in the game world.



Source: http://critical-gaming.squarespace.com/blog/2010/12/30/about-that-indie-feel-pt1.html

Some gamers will complain that the hit box in their game is not the right size – either it is too large or too small for the character. In this assignment, we are going to calculate the four coordinates of a hit box based on user input. If you ever build a video game, you can use this program to help you calculate your own hit boxes!

Your task is to:

A) Prompt the user to enter an X coordinate, Y coordinate, Width, and Height
B) Read in those four values
C) Perform calculations to determine the four X, Y coordinates of the hit box
D) Store those results in separate variables
E) Display the result to the user

The algorithm output is as shown below, with user input in bold. Follow the output format exactly. Save your source code in a file called **Assignment2B** (with a file extension of **.cpp**, **.cs** or **.java**)

Sample Output:

```
Enter the hit box bottom-left X coordinate: 3
Enter the hit box bottom-left Y coordinate: 7
Enter the width of the hit box: 4
Enter the height of the hit box: 7
[Hit Box Coordinates]
Bottom-Left: 3, 7
Top-Left: 3, 14
Bottom-Right: 7, 7
Top-Right: 7, 14
```

Assignment 2C:

*Currency Conversion***:** You are planning a trip to the far away land of Bisonica. The currency of Bisonica is the Bison Dollar. You do not remember the conversion rate between US Dollars and Bison Dollars, but you do know that a Bison Dollar is worth 5 British Pounds.

At the time of this writing, a US Dollar is worth 0.79 British Pounds. For this assignment, create a conversion program that prompts the user to enter how many US Dollars they have. Then, provide the conversion to British Pounds and then to Bison Dollars.

Important Note: You must take in the US Dollar amount as an **integer**, and then appropriately handle the type casting during the currency conversion. Additionally, you must use **constants** to store the conversion value for both British Pounds and Bison Dollars.

The algorithm output is as shown below, with user input in bold. Follow the output format exactly. Save your source code in a file called **Assignment2C** (with a file extension of **.cpp**, **.cs** or **.java**)

Sample Output:

```
Welcome to Bisonica!
How many US Dollars do you have? 10
10 US Dollar(s) is 7.9 British Pound(s),
which is 1.58 Bison Dollar(s)!
```

Submission:

1. You will submit 3 separate files – one for each of the assignments above.
2. Upload separate files (one for each assignment) to the assignment submission folder in Gradescope. Do NOT submit homework in D2L.
3. Submit your work by the due date.