CS 010 - Introduction to Computer Science I

Lab 3 - Integer Arithmetic and Switching Data Types

**Suggested Pre-Lab Work (you should have completed at least some of these items)**

Zyante Chapter 2, and corresponding Codelab exercises

Video tutorials: [Module 3 playlist](http://www.youtube.com/playlist?list=PLTTJbxrH72A1RyceTFxRdBcy-9J8wFqRH)

**Collaboration policy**

Collaboration on Piazza, Cloud9 or in person on these lab exercises is strongly ENCOURAGED. They are intended for practice, not assessment -- feel free to ask for help from, and provide help to, others. **You may not, of course, blindly copy solutions from one another (or from anywhere else) or simply write code for someone else,** but you can certainly help each other debug, give plenty of suggestions and hints, *explain* why things work or don't work, etc.

Read the full policy at: [Full Collaboration Policy](https://docs.google.com/document/d/1WyzL3qvKLrC1UCRf178b_wYWQmEZlhDObFNFb79U63I/edit?usp=sharing)

**Lab Objectives**

To gain experience with:

|  |  |  |  |
| --- | --- | --- | --- |
| * integer variables | * integer math | * integer division | * mod operator |

**Exercise 0: Setup & Lab Submission Framework**

You will continue to **submit a single file to R'Sub**, but you will follow the framework below to set up your program.

Your source code file will have multiple blocks, one for each exercise. The framework allows the tester (human or R'Sub) to specify the exercise to test during program execution. At this point we ask you to simply utilize the framework, not fully understand branches. **All the code for an exercise - everything you would previously have put inside main - will now go in the corresponding code block (see comments below).**

|  |  |
| --- | --- |
| **Framework** | **Setup** |
| #include <iostream>  using namespace std;  int main()  {  int ex;  cout << "Which exercise? ";  cin >> ex;  cout << endl;  if ( ex == 1 )  {  // All Exercise 1 code  }  else if ( ex == 2 )  {  // All Exercise 2 code  }    return 0;  } | 1. Create a file named lab3.cpp within the proper directory. 2. Copy and paste the denoted framework into your file. 3. Copy-paste the proper assessment header, then fill in. |

**Exercise 1: Integer Mathematics**

Write a program that accepts 2 integer values from the user. The prompts prior to input should match the examples. After acquiring the two inputs, your program should calculate and output the sum, product, integer quotient, and modulus in exactly the same format as the examples. Make sure your solution has the same vertical spacing (blank lines) as the examples.

**Think About the Problem**

1. Based on the description of the problem, what are the tasks the program must do?
   1. each task should be a comment
   2. recall comments start with //
   3. each comment should be an English description of the task, not C++ code
2. Certain tasks may require storage of values, we store values in variables.
   1. Take a moment to identify how many items you wish to store, based on your tasks.
   2. Create a comment with an English description for each item you wish to store.
   3. Now, think about the type of variable required to store each of the identified items.
3. Now, within your comments, you have a clearly outlined program.

**Program Your Solution**

1. Now, on the line following each of your comments, write the C++ code that performs that task: this allows a reader to read and think about the comment, and then check (or skip) the actual implementation.
2. You will often need to to check an example program run to get your output to be as expected.

**Example Executions** (User input has been **bolded and underlined** to help differentiate typed input from program output.)

|  |  |
| --- | --- |
| **Example 1:**  Enter the first integer: **10**  Enter the second integer: **25**  10 + 25 = 35  10 \* 25 = 250  10 / 25 = 0  10 % 25 = 10 | **Example 2:**  Enter the first integer: **15**  Enter the second integer: **5**  15 + 5 = 20  15 \* 5 = 75  15 / 5 = 3  15 % 5 = 0 |
| **Example 3:**  Enter the first integer: **12**  Enter the second integer: **10**  12 + 10 = 22  12 \* 10 = 120  12 / 10 = 1  12 % 10 = 2 |  |

**Exercise 2: Cash Register**

Your goal is to simulate a cash register. Your program should first ask for the purchase amount and then the amount paid by the customer. Then the program should calculate the total change and the quantities of each monetary denomination to arrive at that change amount. Finally, the program should output the total change followed by how many of each type of currency to give back to the customer.

***Algorithm outline:*** *Calculate the floating point change value. Next convert the entire change amount to the amount of pennies (if all the change was given in pennies), assigning the number of pennies into a new variable. Then use* ***integer*** *division to calculate the number of each coin denomination, and use the* ***modulo*** *operator to* ***update*** *the remaining number of pennies each time.*

The program requires that we convert from floating point data (change in dollars) to integer data (change in pennies) through a process called *casting.* Specifically, we use an operator called a static\_cast as follows:

int\_variable = static\_cast<int>( fp\_variable );

We have to be careful about this: it doesn’t always give us the result we expect!

Since it *truncates* (rather than rounding), a number such as 46.999999 will become **46** as an int. This is problematic because 47.0 could be stored by the computer as 46.999999 or 47.000001, leading to different results!

To fix these divergent results, we add **0.5** before casting, forcing the number to have the expect integer value.

x = static\_cast<int>(y + 0.5);

In our case, we must calculate the number of pennies before casting, so we alter the above to get:

x = static\_cast<int>(y \* 100 + 0.5);

**For this lab exercise, what are good variable names for x and y?   
What data types should each variable be?**

**Think About the Problem**

1. Based on the description of the problem, what are the tasks the program must do?
   1. each task should be a comment
   2. recall comments start with //
   3. each comment should be an English description of the task, not C++ code
2. Certain tasks may require storage of values, we store values in variables.
   1. Take a moment to identify how many items you wish to store, based on your tasks.
   2. Create a comment with an English description for each item you wish to store.
   3. Now, think about the type of variable required to store each of the identified items.
3. Now, within your comments, you have a clearly outlined program.

**Program Your Solution**

1. Now, on the line following each of your comments, write the C++ code that performs that task: this allows a reader to read and think about the comment, and then check (or skip) the actual implementation.
2. You will often need to to check an example program run to get your output to be as expected.

**Example Executions** (User input has been **bolded and underlined** to help differentiate typed input from program output.)

|  |  |
| --- | --- |
| Which exercise? **2**  Enter purchase amount: **18.89**  Enter amount received: **20.00**  Total Change: $1.11  dollars 1  quarters 0  dimes 1  nickels 0  pennies 1 | Which exercise? **2**  Enter purchase amount: **25.56**  Enter amount received: **40**  Total Change: $14.44  dollars 14  quarters 1  dimes 1  nickels 1  pennies 4 |

**Exercise 3: Survey**

Please complete the linked survey. Completion should take no more than 15 minutes. <http://goo.gl/VSKA5u>