Trent University
COIS1020H
Lab 1 (Windows Version)

1) Getting familiar with MS Visual C# IDE

By now you should have downloaded your version of Microsoft Visual C# Express 2010 onto your computer (it is available for download the BlackBoard Learn course page). If you have not (and there is no time for doing this in the lab), or you do not have it with you, please use one of the lab computers. The goal of the first question is to simply get you used to the software and the submission process.

For this question, you are required to create and run a simple program (see below), demonstrate that it works to the lab personnel and then submit your .cs file to the Lab 1 dropbox on BlackBoard Learn.

*** It is important that you demonstrate the working program to the Lab personnel or you will not receive marks for this portion of the lab.

The files to submit can most likely be found in your Documents (or My Documents) folder in the following locations.

The **.cs** file is in:
Visual Studio 2010\Projects*ProjectName**ProjectName**filename.cs*
where *ProjectName* and *filename* are the ones that you choose

Here is the program (ignore the line number and change xxx to your name) ...

```
1. using System;
2. public static class Lab1_1
3. {
4.    public static void Main()
5.    {
6.       Console.WriteLine("Hello COIS1020H, my name is xxx");
7.       Console.ReadLine();
8.    }
9. }
```

To create a program in Microsoft Visual C#, follow these steps:
1) Start the Visual C# application
2) Click on **New Project** button (Ctrl+Shift-N).
   - All programs need a project container to run
3) Choose the **Empty Project** template as it creates the least amount of troubles
4) Enter a project name (Lab1_1)
5) Click on **Add New Item** button (Ctrl+Shift+A).
6) Choose the **Code File** template (not the **Class** template as it creates unnecessary overhead)
   - Projects can have many Code Files in them

7) Enter a file name (Lab1_1)
8) Enter the above program using the editor
9) To compile the program, click **Debug** on the menu bar, and then click **Build Solution** (F6).
10) To run the program, click **Debug** on the menu bar and then click **Start Debugging** (F5).
    - You could also press Ctrl+F5 to run the program and it would skip the debugging
11) The `Console.ReadLine();` is added to the end of the program so that the Output Window does not disappear (this will occur when you run with F5 and not Ctrl+F5)
12) Once you are sure that your program works properly, select the **Save All** button (Ctrl+Shift+S) to save all the project files (which included the Lab1_1.cs and Lab1_1.exe files)
13) Upload your Lab1_1.cs file to the Lab 1 DropBox on Blackboard.

2) Larger Example: Input/Processing/Output

a) For this question, you will need to create a brand new project (name it Lab 1_2) with a new code file and then enter the program below (ignore the line numbers).

Please note that there is a text version of this program named Lab1_2.txt (without the line numbers) available in the Labs folder on BlackBoard for copying and pasting. Also, please be aware that the line numbers in the lab exercise may not be the same as the line numbers you see in Visual Studio.

Answer all the highlighted questions in a file and then submit a PDF of this file (called it Lab1_2.pdf) to the Lab 1 dropbox. When asked "What is the output", simply type in what is seen in the output window.

```
1.      public static class Lab1_2
2.      {
3.          public static void Main()
4.          {
5.              int idNum;
6.              double payRate, hours, grossPay;
7.              string firstName, lastName;
8.
9.              // prompt the user to enter employee's first name
10.             Console.Write("Enter employee's first name => ");
11.             firstName = Console.ReadLine();
12.
13.             // prompt the user to enter employee's last name
14.             Console.Write("Enter employee's last name => ");
15.             lastName = Console.ReadLine()
16.
17.             // prompt the user to enter a six digit employee number
18.             Console.Write("Enter a six digit employee's ID => ");
19.             idNum = Convert.ToInt32(Console.ReadLine());
20.
21.             // prompt the user to enter the number of hours employee worked
22.             Console.Write("Enter the number of hours employee worked => ");
23.             hours = Convert.ToDouble(Console.ReadLine());
24.
25.             // prompt the user to enter the employee's hourly pay rate
26.             console.Write("Enter employee's hourly pay rate: ");
27.             payRate = Convert.ToDouble(Console.ReadLine());
28.
29.             // calculate gross pay
```

```
30.             grossPay = hours * payRate;
31.
32.             // output results
33.             Console.WriteLine("Employee {0} {1}, (ID: {2}) earned {3}",
34.                       firstName, lastName, idNum, grossPay);
35.
36.             Console.ReadLine();
37.         }
38.     }
```

b) Build the solution (F6). Resolve any syntax mistakes until the program compiles. There are 3 intentional mistakes already (think System, semicolon and output), plus whichever ones you may make while copying. The same mistake may lead to cascading error messages.

What are the 3 intentional mistakes?

Run the program (F5) and when prompted using this input:

First name	John
Last name	Smith
ID	123456
Hours worked	22.5
Pay rate	14.85

What is the output?

c) In Line 33, replace {3} with {3:C} and run the program again.

What is the output? What is the difference in the output between this and the previous run?

d) Replace the statement on Lines 33-34 with the following statement:

```
Console.WriteLine("Employee {0} {1}, (ID: {2}) earned {3:F4}",
      firstName.ToUpper(), lastName.ToLower(), idNum, grossPay);
```

Build the solution. Resolve any syntax mistakes (if you make any) until the program compiles. Execute it with the provided input.

What is the output? What is the difference in the output between this and the previous run?

e) Return Lines 33-34 to the form:

```
Console.WriteLine("Employee {0} {1}, (ID: {2}) earned {3:C}",
      firstName, lastName, idNum, grossPay);
```

Assume employees have their wages reduced by 20% to reflect the effects of taxes. Add a new line of code into the program to compute `netPay` which is `grossPay` multiplied by 0.8 (80%). Please note that you will have to modify the variable declarations in Line 6 to add `netPay` and as well modify Lines 33-34 to print out the value of `netPay` as well as `grossPay`.

Show the new line of code that was added as well as the modified variable declaration and output statement.

What is the output?

f) Assume that instead of hardcoding the tax rate at 20%, you want the user to input this value. Add a new variable called `taxRate` to Line 6 and then, as was done in Lines 25-27, have the user input a value to `taxRate`. Finally, modify the line of code where `netPay` is computed to use `taxRate` as opposed to 0.8.

Show the modified lines of code.

What is the output (assuming a tax rate of 25%)?

g) One last task to complete the lab. For this part, you are to modify the output statement (Lines 33-34) to print out the data in a different format. Assume we want to print out just the last name, id number and net pay (e.g. Smith: 123456 => $250.59).

Show the modified lines of code.

What is the output (assuming the same input as in Part (f)?

Once you have completed all the questions for Part 2, put the results into a PDF file (use Microsoft Word and Export as a PDF) and then submit the pdf to the Lab 1 Dropbox.

3) Putting it All Together

For the last part of the lab, you are required to complete the following program that computes the amount of money in a piggy-bank based on the number of each type of coin (pennies, nickels, dimes, quarters, loonies, and twoonies). The program prompts the user for their name and the number of each type of coin. The program then computes the amount in the piggy bank and prints the result in an appropriate fashion (dollar sign and two digits after the decimal point).

Enter the following program into C# (ignore the line numbers) and replace (or complete) the lines marked with // ******* with the appropriate C# code (may require more than one line of code to complete the missing parts). Please note that there is a text version of this program named Lab1_3.txt (without the line numbers) available in the Labs folder on BlackBoard for copying and pasting. Also, please be aware that the line numbers in the lab exercise may not be the same as the line numbers you see in Visual Studio.

```
1.  // Name: xxxxxxxxxxxxx
2.  // Student Number: xxxxxxx
3.  // Lab 1, Part 3
4.  // Program Description: This program computes the amount of money in a
5.  //     piggy-bank given the number of pennies, nickels, dimes, quarters,
6.  //     loonies, and twoonies).
7.
8.  using System;
9.  public static class Lab1_3
```

```
10. {
11.     public static void Main()
12.     {
13.     // declare the variables and constants
14.     int numPens, numNicks, numDimes, numQuarts, numLoons, numTwoons;
15.     double amtInPiggyBank;
16.     string name;
17.
18.     // Input the name of the user
19.     Console.Write("Enter your name => ");
20.     name = Console.ReadLine();
21.
22.     // Input the number of pennies
23.     Console.Write("Enter the number of pennies in the Piggy Bank => ");
24.     numPens = Convert.ToInt32(Console.ReadLine());
25.
26.     // Input the number of nickels
27.     Console.Write("Enter the number of nickels in the Piggy Bank => ");
28.     numNicks = Convert.ToInt32(Console.ReadLine());
29.
30.     // Input the number of dimes
31.     //*** Insert code in input the number of dimes
32.
33.     // Input the number of quarters
34.     //*** Insert code in input the number of quarters
35.
36.     // Input the number of loonies
37.     //*** Insert code in input the number of loonies
38.
39.     // Input the number of twoonies
40.     //*** Insert code in input the number of twoonies
41.
42.     // Compute the amount in the Piggy Bank
43.     //*** Complete the following to include the remaining types of coins
44.     amtInPiggyBank = numPens * 0.01 + numNicks * 0.05;
45.
46.     // Print out the amount in the Piggy Bank
47.     //*** Fix the following statement so it prints a dollar sign
48.     //*** and two digits after the decimal point (eg. $12.65)
49.     Console.WriteLine("{0} has {1} in the Piggy Bank", name, amtInPiggyBank);
50.     Console.ReadLine();
51.     }
52. }
```

Once you are comfortable that the program works correctly, demonstrate it to the lab personnel, and then submit your Lab1_3.cs file to the Lab 1 dropbox.