Trent University
COIS1020H
Lab 3 (Windows Version)

1) <u>Introduction to Methods (with Debugging)</u>

In the first part of this lab, we would like to introduce you to methods, parameter passing, and the helpfulness of the Debugger.

a) Enter the following program (use Lab3_1 for both the Project and Code File names). This program prints out a corresponding letter grade for a given mark. It uses a method (called *MarktoGrade*) to determine the letter grade. *MarktoGrade* takes one integer call by value formal parameter (called *mark*) and returns a char value that is the letter grade. All of the input and output (except an error message) is done in *Main*. There is one syntax error in this program that you will need to fix before it will compile (the error is in the line where the method is called).

   Please note that there is a text version of this program named Lab3_1.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. using System;
2. public static class Lab3_1
3. {
4.     public static void Main()
5.     {
6.         // declare variables
7.         int inpMark;
8.         string lastName;
9.         char grade = ‘ ‘;
10.
11.         // enter the student's last name
12.         Console.Write("Enter the last name of the student => ");
13.         lastName = Console.ReadLine();
14.         // enter (and validate) the mark
15.         do {
16.             Console.Write("Enter a mark between 0 and 100 => ");
17.             inpMark = Convert.ToInt32(Console.ReadLine());
18.         } while (inpMark < 0 || inpMark > 100);
19.
20.         // Use the method to convert the mark into a letter grade
21.         grade = MarkToGrade();
22.
23.         // print out the results
24.         Console.WriteLine("{0}'s mark of {1} converts to a {2}", lastName,
25.         inpMark, grade);
26.         Console.ReadLine();
27.     }
28.
29.     // Method: MarkToGrade
30.     // Parameter
```

```
    //      mark: call by value parameter to be converted to a letter grade
    // Returns: the corresponding letter grade
    public static char MarkToGrade(int mark)
    {
        char letterValue;

        // multi-alternative If statement to determine letter grade
        if(mark > 0 && mark < 50)
            letterValue = 'F';
        else if(mark >= 50 && mark < 60)
            letterValue = 'D';
        else if(mark >= 60 && mark < 75)
            letterValue = 'C';
        else if(mark >= 75 && mark < 85)
            letterValue = 'B';
        else if(mark >= 85 && mark <= 100)
            letterValue = 'A';
        else
        {
            Console.WriteLine("***Error - invalid mark");
            letterValue = 'X';
        }

        //return the letter grade back to Main
        return letterValue;
    }
}
```

b) Run the program using input Smith and 98

What is the syntax error and how did you fix it?

What is the output?

c) Run the program using input Smith and 34

What is the output?

d) Run the program using input Smith and 59

What is the output?

e) Run the program using input Smith and 72

What is the output?

f) Run the program using input Smith and 83

What is the output?

g) Run the program using input Smith and then -1, 101 and 90 for the marks

What is the output?

Why do you think the input values for weight in Parts (b)-(g) were chosen?

h) Now we are going to see how the Debugger can help us understand the functionality of methods. Put in a Breakpoint (F9) at the point where the method is invoked or called (Line 21). You will see that the line is highlighted in red with a red dot to the left of the line.

Now run the program (F5) using a last name of Smith and a mark of 75. Notice that the program stops at the line with the Breakpoint (and the line is now highlighted in yellow). Also notice that in the bottom left part of Visual Studio there is a window labeled *Locals* that shows the current values for all the variables declared in `Main()`: `inpMark, lastName,` and `grade`. Press (F11) once and notice the program has jumped into the method *MarkToGrade* and the values in the *Locals* window now correspond to the variables and parameters of the method: `mark` and `letterValue`. Also notice that in the lower right hand side window (*Call Stack*), there are two entries: one for *Main* and one for *MarkToGrade* with the later one active (yellow arrow). Continue to single step your program until you reach the `return` statement in Line 55.

Single step from Line 55 and describe what you are see in terms of where you are in the program (line number), what is in the *Locals* window (variables and values), and what is in the *Call Stack* window (will require two to three more single steps to see the program jump.

i) Remove the BreakPoint from Line 21: press (F9). At Line 53, add the following statement:

```
        Console.WriteLine("The value of the mark is {0}", inpMark);
```

What error message did you receive and why?

j) Remove the extra line from Part (i). Now let's change the method header to include a call by address (reference) parameter. Change Line 33 to be:

```
        public static void MarkToGrade(int mark, ref char letterValue)
```

Remove Lines 35 and 55

```
        char letterValue;
        return letterValue;
```

Change Line 21 to be:

```
        MarkToGrade(inpMark, ref grade);
```

Run the program with the input of Smith and 98.

What is the output?

What do you notice about the output relative to what you saw in Part (b)?

Why did we have to remove Line 35 (try putting it back in)?

What did we have to remove Line 55 (try putting it back in)?

Why did the method call (invocation) in Line 21 have to change?

k) One last set of changes to the program from Part (j). In the method call (Line 21), change it to

```
   grade = MarkToGrade(inpMark);
```

What error message did you receive and why?

What if we change Line 21 to: `grade = MarkToGrade(inpMark, ref grade);`

What error message did you receive and why?

Finally, how about if we change Line 21 to: `MarkToGrade(ref grade, inpMark);`

What error messages did you receive and why?

Answer all the highlighted questions in a file and then submit a PDF of this file (called it Lab3_1.pdf) to the Lab 3 dropbox. When asked “What is the output”, simply type in what is seen in the output window.

2) Putting it All Together

a) Here is a program that uses two user-defined methods: one with call by value formal parameters and one with call by address formal parameters. The program computes the Gross Pay and Net Pay for an employee after entering the employee’s last name, hours worked, hourly rate, and percentage of tax. Enter the following program into C# (ignore the line numbers) and replace 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 Lab3_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.

```
1. // Name: xxxxxxxxxxxxx
2. // Student Number: xxxxxxx
3. // Lab 3, Part 2
4. // Program Description: This program uses two user defined methods to compute
5. //    the gross pay and net pay for an employee after entering the employee’s
6. //    last name, hours worked, hourly rate, and percentage of tax.
7.
8.  using System;
9.  public static class Lab3_2
10. {
11.    public static void Main()
12.    {
13.       // declare variables
14.       int hrsWrked;
15.       double ratePay, taxRate, grossPay, netPay=0;
16.       string lastName;
17.
18.      // enter the employee's last name
19.      Console.Write("Enter the last name of the employee => ");
20.      lastName = Console.ReadLine();
21.
22.      // enter (and validate) the number of hours worked (positive number)
23.      do
```

```
     {
        Console.Write("Enter the number of hours worked (> 0) => ");
        hrsWrked = Convert.ToInt32(Console.ReadLine());
     } while (hrsWrked < 0);

     // enter (and validate) the hourly rate of pay (positive number)
     // *** Insert the code to enter and validate the ratePay

     // enter (and validate) the percentage of tax (between 0 and 1)
     // *** Insert the code to enter and validate taxRate

     // Call a method to calculate the gross pay (call by value)
     grossPay = CalculateGross(hrsWrked, ratePay);

     // Invoke a method to calculate the net pay (call by reference)
     CalculateNet(grossPay, taxRate , ref netPay);

     // print out the results
     Console.WriteLine("{0} worked {1} hours at {2:C} per hour", lastName,
                       hrsWrked, ratePay);
     // *** Insert the code to print out the Gross Pay and Net Pay
     Console.ReadLine();
  }

  // Method: CalculateGross
  // Parameters
  //      hours: integer storing the number of hours of work
  //      rate: double storing the hourly rate
  // Returns: double storing the computed gross pay
  public static double CalculateGross(int hours, double rate)
  {
     // *** Insert the contents of the CalculateGross Method
  }

  // Method: CalculateNet
  // Parameters
  //      grossP: double storing the grossPay
  //      tax: double storing tax percentage to be removed from gross pay
  //      netP: call by reference double storing the computed net pay
  // Returns: void
  public static void CalculateNet(double grossP, double tax, ref double netP)
  {
     // *** Insert the details of the CalculateNet Method
  }
}
```

b) Build the solution. Resolve any syntax mistakes (if you make any) until the program compiles. Run the program using the input data: Smith, 40, 12.5, 0.2. You should find that the gross pay is $500.00 and net pay is $400.00

What is the output?

c) Run the program using the input data: Smith, -8, 40, 12.5, 0.2

What is the output?

d) Run the program using the input data: Smith, 40, -3, 12.5, 0.2

What is the output?

e) Run the program using the input data: Smith, 40, 12.5, 9, 0.2

What is the output?

f) Run the program using the input data: Smith, 44, 12.5, 0.2

What is the output?

g) Now let's add another user-defined method to our program … this method will compute any overtime pay (assume the employees receive time and a half (1.5) for any hours worked over 40). Your method is to invoked after you compute gross pay with *CalculateGross* but before you compute net pay with *CalculateNet* (i.e., between Lines 36 and 39). The new method is to be called *CalculateOT*, and it is to have to following method header:

```
public static double CalculateOT(int hours, double rate)
```

where both formal parameters are call by value with *hours* representing the number of hours work and *rate* representing the hourly wage. *CalculateOT* is to return only the amount of overtime pay (in excess of regular pay) which is computed from the number of hours greater than 40 multiplied by the hourly wage and then multiplied by 0.5. Please note that *CalculateOT* is only returning the overtime amount which must be added to the gross pay variable in *Main()*.

Run the program using the input data: Smith, 40, 12.5, 0.2

What is the output?

h) Run the program using the input data: Smith, 44, 12.5, 0.2

What is the output?

How does the result differ from Part (f)?

i) Run the program using the input data: Smith, 20, 12.5, 0.2

What is the output?

Once you are comfortable that the program works correctly, demonstrate it to the lab personnel, answer all the highlighted questions in a file and then submit your Lab3_2.pdf and Lab3_2.cs Lab3_2.exe files to the Lab 3 dropbox