Trent University
COIS1020H
Lab 4 (Windows Version)

1) Introduction to Arrays (with Debugging)

In the first part of this lab, we would like to introduce you to arrays, parameter passing, and the more uses of the Debugger.

a) Enter the following program (use Lab4_1 for both the Project and Code File names). This program focuses on arrays and uses two user-defined methods: one to have the user fill up an array and the second to print out the contents of an array. Enter the program into Microsoft Visual C# (ignore the line numbers). Notice again how we simplify *Main()* and leave much of the real work to the user-defined methods. There is one syntax error in this program that needs to be fixed before it will compile.

Please note that there is a text version of this program named Lab4_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 Lab4_1
3. {
4.     public static void Main()
5.     {
6.         int[] dataArray = new int[10] {9, 8, 7, 6, 5, 4, 3, 2, 1, 0};
7.         int n = 0;
8.
9.         // Fill the array
10.         Fill(dataArray, ref n);
11.
12.         // Display the array contents
13.         Dump();
14.
15.         // Pause until user is done
16.         Console.ReadLine();
17.     }
18.
19.     // Method:       Fill
20.     // Description:  Fill an array with values.
21.     // Parameters:   arrValues: the array to fill.
22.     //               num: number of elements in the array.
23.     // Returns:      void
24.     public static void Fill(int[] arrValues, ref int num)
25.     {
26.        // input the number of data values to put in the array
27.        do
28.        {
29.            Console.Write("Enter the number of elements (between 0 and 10) => ");
30.            num = Convert.ToInt32(Console.ReadLine());
31.        } while(num < 0 || num > 10);
32.
33.        // loop to enter the values
```

```
34.         for(int i = 0; i < num; ++i)
35.         {
36.            Console.Write("Enter the Element {0} => ", i);
37.            arrValues[i] = Convert.ToInt32(Console.ReadLine());
38.         }
39.      }
40.
41.      // Method:       Dump
42.      // Description:  Display the contents of an array.
43.      // Parameters:   arrVals: array to be displayed.
44.      //               num: number of elements in the array.
45.      // Returns:      void
46.      public static void Dump(int[] arrVals, int num)
47.      {
48.         // output the values from the array
49.         Console.WriteLine("The elements in the array are:");
50.         for (int i = 0; i < num; ++i)
51.            Console.WriteLine(" {0}", arrVals[i]);
52.      }
53.  }
```

b) Run the program using input 4, -7, 29, 8, 34

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

What is the output?

c) Run the program using input -2, 15, 4, -7, 29, 8, 34

What is the output?

Why is it the same as in Part (b) even though the input is not the same?

d) Now we are going to again investigate the usefulness of the Debugger. Put in two Breakpoints (F9): one at the point where the method *Fill* is invoked (Line 10) and one where *Dump* is invoked (Line 13). You will see both lines are highlighted in red with a red dot to the left of the line.

e) Now run the program (F5) using the input of 4, -7, 29, 8, 34. Notice that the program stops before you enter any input. This is due to the fact that the program has not invoked the method where the data is input (*Fill*). Look at the *Locals* window at the bottom left part of Visual Studio. Notice that there are two identifiers in this window: `dataArray` and `n`. The identifier `dataArray` has a "+" preceding it indicating that it is a complex data type that contains additional information (`n` is a scalar (or simple) data type that contains the number 0).

Click on the "+" and you will see the contents of `dataArray` (the numbers 9 – 0 that the array was initialized with). Press (F11) once and notice the program has jumped into the method `Fill` and the values in the *Locals* window now correspond to the variables and parameters of the method: `arrValues` and `num`.

What are the values in `arrValues` and `num` in *Fill*?

f) Press (F5) to resume your program from the current instruction and notice that it provides you the opportunity now to input the data. Once you are done inputting the data (4, -7, 29, 8, 34), the program will continue to the next breakpoint or to the end if there are not more breakpoints. In your case, the program will stop at the second breakpoint on Line 13 (call to *Dump*). From the *Locals* window examine `dataArray` and `n`.

What are the values in `dataArray` and `n` in *Main*?

Why have they changed?

g) Press (F11) once again and notice the program has jumped into the method *`Dump`* and the values in the *Locals* window now correspond to the variables and parameters of the method: `arrVals` and `num`.

What are the values in `arrVals` and `num` in *Dump*

h) Single step (F11) your way through *Dump* until at least 2 values from the array are printed on the output screen and then press (F5) to move to the end of the program.

What is the output?

Why are only 4 values from the array being output given that you know from Part (g) that there were 10 values in the array that was to be printed?

i) Remove the breakpoints from Lines 10 and 13. Let's add one more user-defined method to the program. This method, called *ReverseDump* is to output the values in the array in revere order (please note that you are NOT to use the *Array.Reverse* method from the *Array* class). The approach is quite simple: your *ReverseDump* method will look very similar to the *Dump* method with the exception that the `for` loop will count backwards from `num-1` to `0`. The method header will look like:

```
public static void ReverseDump(int [] arrItems, int num)
```

Give the details of the *ReverseDump* method.

Give the details of the *ReverseDump* call (in *Main*).

What is the output?

Answer all the highlighted questions in a file and then submit a PDF of this file (called it Lab4_1.pdf) to the Lab 4 dropbox. When asked "What is the output", simply type in what is seen in the output window.

2) Putting it All Together

a) A very common application of arrays is computing statistics on lists of numbers. Below you will find a template of a program that uses four user-defined methods: input an array of numbers, compute the average the array, find the largest number in the array, and find the smallest number in the array. 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 Lab4_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 4, Part 2
4. // Program Description: This program uses four user-defined methods: one
5. //    to input an array of numbers, one to compute the average the array, one
6. //    to find the largest number in the array, and one to find the smallest
7. //    number in the array.
8.
9.  using System;
10.  public static class Lab4_2
11. {
12.    public static void Main()
13.    {
14.       int[] compStats = new int[25];
15.       int n = 0, large, small;
16.       double avg = 0;
17.
18.       // Input values into the array
19.       InpArray(compStats, ref n);
20.
21.       // Find the average of the elements in the array
22.      // *** Insert the code for the call to the FindAverage method
23.
24.       // Find the largest element in the array
25.      // *** Insert the code for the call to the FindLarge method
26.
27.       // Find the largest element in the array
28.      // *** Insert the code of the call to the FindSmall method
29.
30.       // Print out the results
31.       Console.WriteLine("\nThe Average of the array is {0:F}", avg);
32.      // *** Insert the code to print out the largest and smallest values
33.
34.       // Pause until user is done
35.       Console.ReadLine();
36.    }
37.
38.    // Method:       InpArray
39.    // Description:  Input values into an array.
40.    // Parameters:   arrValues: the array to fill.
```

```
41.   //               num: number of elements in the array.
42.   // Returns:      void
43.   public static void InpArray(int[] arrValues, ref int num)
44.   {
45.      // input the number of data values to put in the array
46.      do
47.      {
48.          Console.Write("Enter the number of elements (<= 25) => ");
49.          num = Convert.ToInt32(Console.ReadLine());
50.      } while (num < 0 || num > 25);
51.
52.      // loop to enter the values
53.      for (int i = 0; i < num; ++i)
54.      {
55.          Console.Write("Enter the Element {0} => ", i);
56.          arrValues[i] = Convert.ToInt32(Console.ReadLine());
57.      }
58.   }
59.
60.   // Method:       FindAverage
61.   // Description:  Computes the average of the elements in the array.
62.   // Parameters:   arrVals: array to be processed
63.   //               num: number of elements in the array.
64.   // Returns:      the average of the elements in the array
65.   public static double FindAverage(int[] arrVals, int num)
66.   {
67.      // *** Insert the details of the FindAverage Method
68.
69.   }
70.
71.   // Method:       FindLarge
72.   // Description:  Determines the largest element in the array.
73.   // Parameters:   arrVals: array to be processed
74.   //               num: number of elements in the array.
75.   // Returns:      the largest element in the array
76.   public static int FindLarge(int[] arrVals, int num)
77.   {
78.      // *** Insert the details of the FindLarge Method
79.   }
80.
81.   // Method:       FindSmall
82.   // Description:  Determines the smallest element in the array.
83.   // Parameters:   arrVals: array to be processed
84.   //               num: number of elements in the array.
85.   // Returns:      the smallest element in the array
86.   public static int FindSmall(int[] arrVals, int num)
87.   {
88.      // *** Insert the details of the FindSmall Method
89.   }
90. }
```

Once you are comfortable that the program works correctly, demonstrate it to the lab personnel, and then submit your Lab4_2.cs file to the Lab 4 dropbox.