

1.1P: Preparing for OOP – Answer Sheet

Introduction

This paper 's answer sheet serves two purposes:

- A. It serves as a revision for you of your previous learnings; and
- B. It establishes a baseline understanding of your knowledge in key Computer Science topics.

As such this paper is divided into the following areas of knowledge:

- A. Your experience with UNIX/DOS console commands;
- B. Your ability to differentiate between data types (e.g. text) and information categories (e.g. title);
- C. Your experience with compiler parsing and evaluation of expressions according to rules of precedence (e.g. BODMAS, also known as GEMS or PEMDAS);
- D. Your understanding of Computer Science concepts and various compiler constructs such as blocks and scope;
- E. Finally taking three steps, we want you to develop a program as follows:
 - 1. starting with a simple function: you provide the pure logic and calculations, no input, nor output;
 - 2. Then, in the second step, you write the main line code that invokes that simple function. Your main line code will provide the necessary data, and then you will print out the result of the function's calculation.
 - 3. Finally we want you to add business logic to the main line program's code; that business logic will interpret the results of the function, and inform your user with information about the results.

Section A: Console commands

- 1. Explain the following terminal instructions:
 - a. `cd`: Change Directory
 - b. `pwd`: Print Working Directory
 - c. `mkdir`: Make Directory
 - d. `cat`: Print out content of a file
 - e. `ls`: List down all the item in the directory

Section B: Data types and Information categories

1. Consider the following categories of information, and suggest the most appropriate data type to store and represent each kind of information:

Information Category	Suggested Data Type
A person's family name	String
A person's age in years	Integer
A person's weight in Kilograms	Float
A telephone number	String
A temperature on the Kelvin scale	Float
The average age of a group of children	Float
Whether the student passed this task	Boolean

2. Aside from the examples already provided above, please come up with your own examples of information that could be stored as:

Data Type	Suggested Information Category
String	Supermarket name
Integer	Number of supermarkets
Float	A person's height in meters
Boolean	Is 2.5 an integer?

Section C: Compiler evaluation of expressions

1. Fill out the **last** two columns of the following table based on the expression and values we have supplied.
2. Evaluate the value of each expression under column 1, given its formula, values, and variables; use the given values (column 2) of any variable(s) in the expression.
3. Identify the value of the results (column 3), and the data type the result is most likely to be (column 4) in a compiler "friendly" form (e.g. Float):

Expression	Given	Result	Data Type
76		76	Integer
True		True	Boolean
a	a = 3.1415927	3.1415927	Float
1 + 2 * 3 + 4		11	Integer
a and False	a = True	False	Boolean
a or False	a = True	True	Boolean
a + b	a = 1 b = 3	4	Integer
3 * a	a = 5	15	Integer
a * 2 + b	a = 2.5 b = 3	8.0	Float
a + 2 * b	a = 2.5 b = 3	8.5	Float
(a + b) * c	a = 2 b = 4 c = 6	36	Integer
"Fred" + " Astair"		Fred Astair	String
a + " Rogers"	a = "Ginger"	Ginger Rogers	String

Section D: Compiler Constructs and CS Concepts:

1. Using some code as an example, please explain the difference between **declaring** and **initializing** a variable.

The difference between the two is: Declaring is used to specify the variables datatype. While initializing is used to assign value to the variable and you can do it with declaration or in a separate line.

Paste your example code below:

```
int x; //Declare
x = 10; //Initialize
int y = 10; //Declare and Initialize
```

2. Explain the term **parameter**. Write some **code** that demonstrates a simple use of a parameter. You should show a procedure or function that uses a parameter, and how you would call that procedure or function.

A parameter is a variable and are used in a function or procedure to do a task that will relate to a data provided as input. A parameter will act as an Argument and it will be passed to the function when it is called.

Paste your example code below:

```
C# HelloWorld_V3 Program
1 using System;
2
3 0 references
4 public class Program
5 {
6     1 reference
7     static void parameter(string name)
8     {
9         Console.WriteLine("Hello " + name + "!");
10    }
11    0 references
12    static void Main(string[] args)
13    {
14        Console.WriteLine("Enter your name: ");
15        string name = Console.ReadLine();
16        parameter(name);
17    }
18 }
```

3. Using an **coding example**, describe the term **scope** as it is used in procedural programming (not in business nor project management). Make sure you explain the differences of as many kinds of scope that you can identify (at least two, and up to five).

Scope is a term to define where variables and functions can be used, scope helps organize and manage the availabilities of variables which will help avoiding unintended error.

Paste your example code below:

Local Scope:

```
using System;

0 references
public class Program
{
    0 references
    static void Main(string[] args)
    {
        int x = 10;
        Console.WriteLine(x);
    }
}
```

Global Scope:

```
using System;

0 references
public class Program
{
    static int x = 10;

    0 references
    static void Main()
    {
        Console.WriteLine(x);
        DifferentFunc();
    }

    1 reference
    static void DifferentFunc()
    {
        Console.WriteLine(x);
    }
}
```

Section E: Implementing Algorithms, Data Handling, and Informing Results - Personalized Requirements

STEP 1:

1. In a procedural style, in any language you prefer, write a function called Average, which accepts an array of integers, and returns the average of those integers.

2. **Do not use any libraries for calculating the average:** we want to see your understanding of algorithms.
3. You must demonstrate appropriate use of parameters, returning and assigning values, and the use of loop(s). **Note — just write the function at this point.** In the next step we will ask you to *invoke the function*.
4. You should **not** have a complete program, **nor** even code that outputs anything at this stage. This is a **function**; and input/output and any business logic processing is the responsibility of the (main line) calling code.

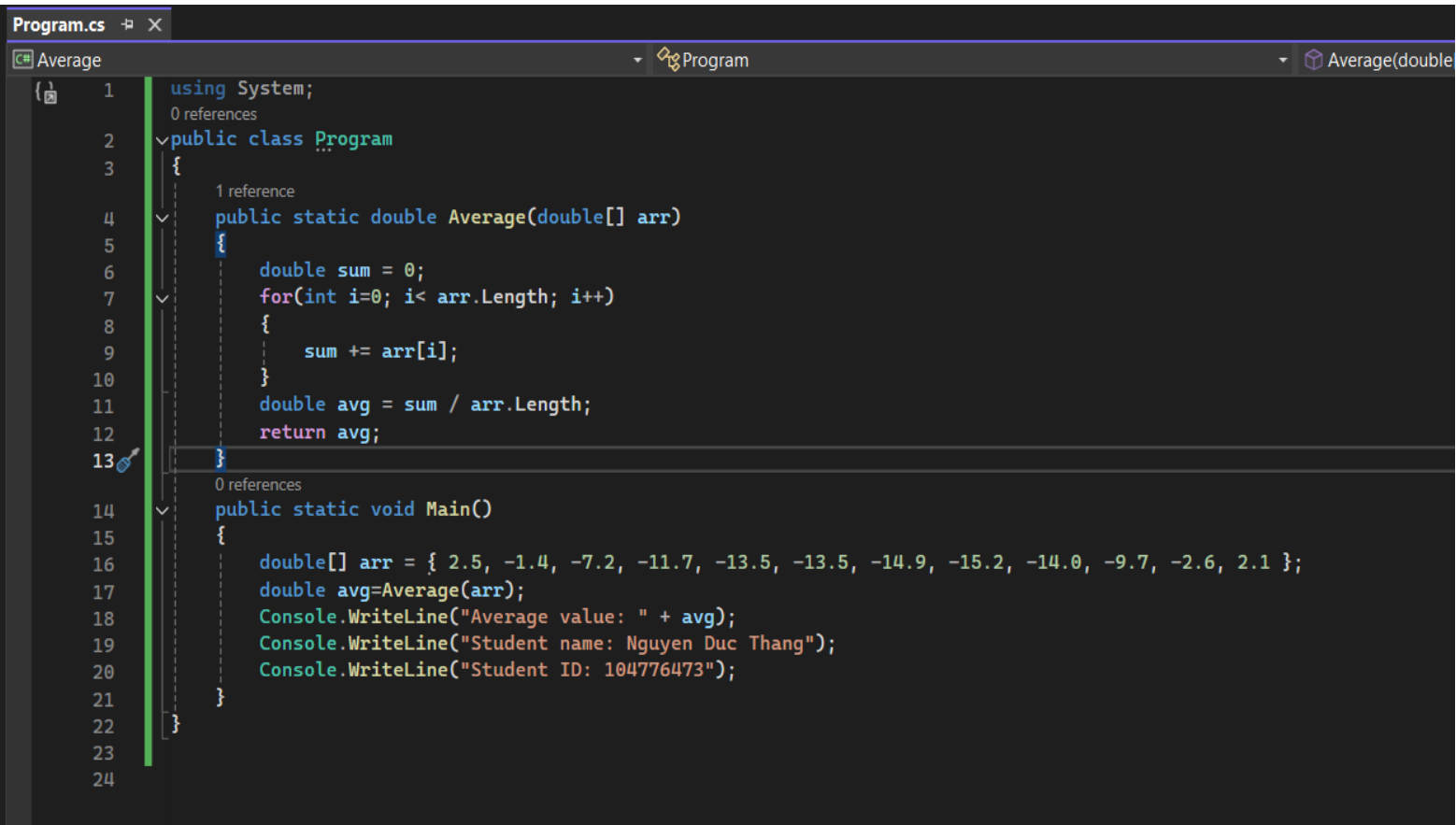
Paste your example function code below:

```
public static double Average(double[] arr)
{
    double sum = 0;
    for(int i=0; i< arr.Length; i++)
    {
        sum += arr[i];
    }
    double avg = sum / arr.Length;
    return avg;
}
```

STEP 2:

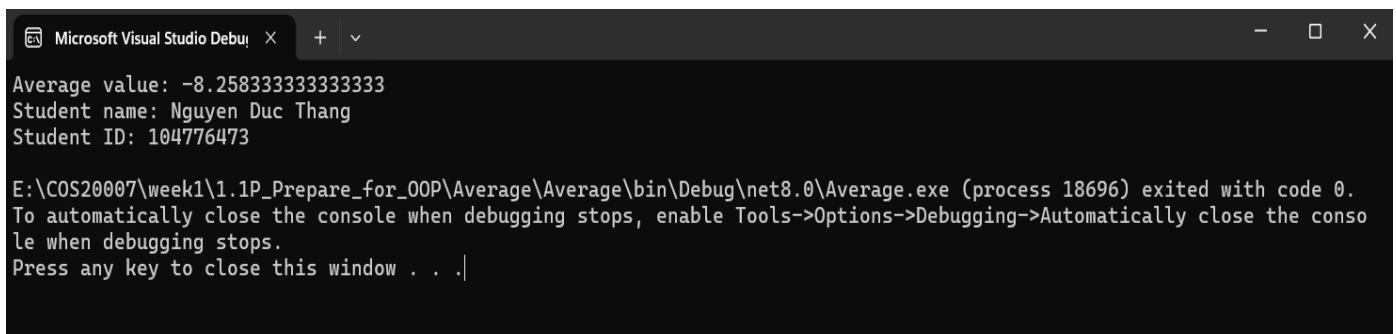
5. Using the same preferred language, write the main line calling code you would need to (a) marshal the data, (b) invoke the function, (c) print out the result, and (d) **print out your student name and student Id**
6. We do **not** require you to provide any input processing logic; you simply have provide the inline instantiate of a collection of data values (provided below) for the function to calculate the average of that data set.
 - a. Sample data values
2.5, -1.4, -7.2, -11.7, -13.5, -13.5, -14.9, -15.2, -14.0, -9.7, -2.6, 2.1
7. Note: you should have made **no changes** to your function.

Paste all of your example code below:



```
1 using System;
2 public class Program
3 {
4     public static double Average(double[] arr)
5     {
6         double sum = 0;
7         for(int i=0; i< arr.Length; i++)
8         {
9             sum += arr[i];
10        }
11        double avg = sum / arr.Length;
12        return avg;
13    }
14    public static void Main()
15    {
16        double[] arr = { 2.5, -1.4, -7.2, -11.7, -13.5, -13.5, -14.9, -15.2, -14.0, -9.7, -2.6, 2.1 };
17        double avg=Average(arr);
18        Console.WriteLine("Average value: " + avg);
19        Console.WriteLine("Student name: Nguyen Duc Thang");
20        Console.WriteLine("Student ID: 104776473");
21    }
22 }
23
24
```

Paste your example code's output here:



```
Microsoft Visual Studio Debug Console
Average value: -8.258333333333333
Student name: Nguyen Duc Thang
Student ID: 104776473

E:\COS20007\week1\1.1P_Prepere_for_OOP\Average\Average\bin\Debug\net8.0\Average.exe (process 18696) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

8. Using the same preferred language, add to your existing main line code above, the following business logic code for interpreting the result of the function's calculations.
9. Print the message "Multiple digits" if the average is above or equal to 10. Otherwise, print the message "Single digits".
10. And then, if the average is negative, add an additional line of output stating "Average value negative".
11. Finally, if the last digit of the average is larger than the last digit of your Student ID, please print the message "**Larger than my last digit**". Otherwise, please print the correct message, either "**Equal to my last digit**" or "**Smaller than my last digit**".
12. Note, you should not have made any changes to your implemented function
13. Provide evidence of your program running, i.e. the code, its environment, and its run time outputs.

```
1 using System;
2 using System.Globalization;
3 public class Program
4 {
5     public static double Average(double[] arr)
6     {
7         double sum = 0;
8         for(int i=0; i< arr.Length; i++)
9         {
10             sum += arr[i];
11         }
12         double avg = sum / arr.Length;
13         return avg;
14     }
15     public static void Main()
16     {
17         double[] arr = { 2.5, -1.4, -7.2, -11.7, -13.5, -13.5, -14.9,
18             -15.2, -14.0, -9.7, -2.6, 2.1 };
19         double avg=Average(arr);
20         Console.WriteLine("Average value: " + avg);
21         Console.WriteLine("Student name: Nguyen Duc Thang");
22         Console.WriteLine("Student ID: 104776473");
23
24         if (Math.Abs(avg)>=10)
25         {
26             Console.WriteLine("Multiple digits");
27         }
28         else
29         {
30             Console.WriteLine("Single digits");
31         }
32
33         if (avg<0)
34         {
35             Console.WriteLine("Average value negative");
36         }
37         string avg_digits = avg.ToString();
38         char last_digit = avg_digits[avg_digits.Length-1];
39
40         if ((double)last_digit-'0' > 3)
41         {
42             Console.WriteLine("Larger than my last digit");
43         }
44         else if ((double)last_digit-'0' == 3)
45         {
46             Console.WriteLine("Equal to my last digit");
47         }
48         else
49         {
50             Console.WriteLine("Smaller than my last digit");
51         }
52     }
53 }
```



```
49
50         Console.WriteLine("Smaller than my last digit");
51     }
52 }
53 }
54
```

Environment:

'Average.exe' (CoreCLR: DefaultDomain): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Private.CoreLib.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'E:\COS20007\week1\1.1P_Prepere_for_OOP\Average\Average\bin\Debug\net8.0\Average.dll'. Symbols loaded.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Runtime.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'c:\program files\microsoft visual studio\2022\community\common7\ide\commonextensions\microsoft\hotreload\Microsoft.Extensions.DotNetDeltaApplier.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.IO.Pipes.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Linq.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Collections.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Console.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Threading.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Runtime.InteropServices.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Threading.Overlapped.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Security.AccessControl.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Security.Principal.Windows.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Security.Claims.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Runtime.Loader.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Collections.Concurrent.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Average.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.7\System.Text.Encoding.Extensions.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

The program '[2868] Average.exe' has exited with code 0 (0x0).

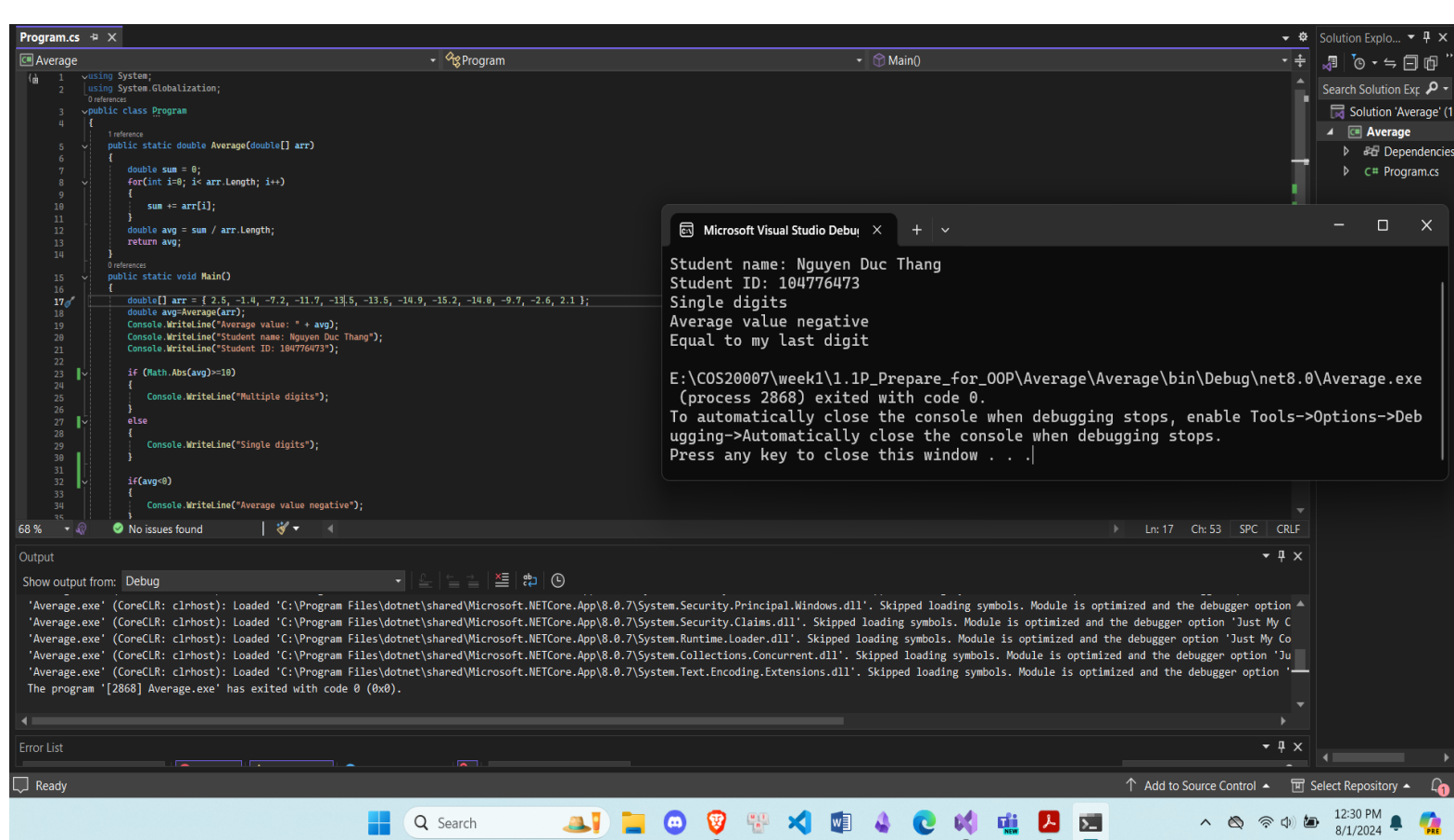
Paste your example code's output here:

```
Microsoft Visual Studio Debug Console

Average value: -8.258333333333333
Student name: Nguyen Duc Thang
Student ID: 104776473
Single digits
Average value negative
Equal to my last digit

E:\COS20007\week1\1.1P_Prepere_for_OOP\Average\Average\bin\Debug\net8.0\Average.exe
(process 2868) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Deb
ugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Finally on a new page paste a SINGLE screenshot of your program (main line and function) running with its outputs here:



End of Task

Please render your paper as a PDF and submit via CANVAS.