1.1P: Preparing for OOP – Answer Sheet

1. Explain the following terminal instructions:
    a. cd: allows the user to go to a directory that they want to access.
    b. ls: shows the collection of files or other directories inside of the directory that the user is currently in.
    c. pwd: shows users all working directories, starting at the root folder.

2. Consider the following kinds of information, and suggest the most appropriate data type to store or represent each:

Information	**Suggested Data Type**
A person's name	String
A person's age in years	Int
A phone number	Int
A temperature in Celsius	Float
The average age of a group of people	Float
Whether a person has eaten lunch	Boolean

3. Aside from the examples already provided in question 2, come up with an example of information that could be stored as:

Data type	**Suggested Information**
String	Title
Integer	PIN number
Float	Travelling distances
Boolean	On and off switch

4. Fill out the last two columns of the following table, evaluating the value of each expression and identifying the data type the value is most likely to be:

Expression	**Given**	**Value**	**Data Type**
6		6	Int

`True`		True	Boolean
`a`	`a = 2.5`	2.5	Float
`1 + 2 * 3`		7	Int
`a and False`	`a = True`	False	Boolean
`a or False`	`a = True`	True	Boolean
`a + b`	`a = 1` `b = 2`	3	Int
`2 * a`	`a = 3`	6	Int
`a * 2 + b`	`a = 2.5` `b = 2`	7	Int
`a + 2 * b`	`a = 2.5` `b = 2`	6.5	Float

`(a + b) * c`	`a = 1`
	`b = 1`
	`c = 5`
		10	Int

`"Fred" + " Smith"`		Fred Smith	String
`a + " Smith"`	`a = "Wilma"`	Wilma Smith	String

5. Using an example, explain the difference between **declaring** and **initialising** a variable.

   When we declare a variable, we set out the standards and characteristics for the existence of that variable such as data type and variable name. Although, the current state of that variable is empty as it doesn't contain any data. After we declare a variable, the variable will be assigned with data either immediately or later during function calls. The data type of the content must be the same as the one that is declared in the variable, or the program will throw a syntax error.

6. Explain the term **parameter**. Write some code that demonstrates a simple of 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 payload of data that is passed on from where the function is being called to the function itself where the data will be processed under the instructions that were written by the developer.

Once the data has been processed, the function will return the output back to where the function is being called and display the results.

```
static int Sum(int x, int y)
{
    return x + y;
}

int total = Sum(10, 20);

Console.WriteLine("Parameters are x=10 and y=10");
Console.WriteLine("The returned sum of both parameters is " + total);
Console.WriteLine("Hello World!");
```

```
Parameters are x=10 and y=10
The returned sum of both parameters is 30
Hello World!
```

7. Using an example, describe the term **scope** as it is used in procedural programming (not in business or project management). Make sure you explain the different kinds of scope.

   Scope is a term in programming that is used to illustrate where a piece of code can be seen or used by other parts of the program. There are two types of scope that are considered the most common types in programming:

   - ***Global scope**: an area that contains code that are visible and can be accessed from any part of the programs. This area is where the program will keep all the data that is required to run the program until the program is closed or exited.*
   - ***Local scope**: an area that contains code that are only accessible and visible to a few certain parts of the program. For instance, local scope is where data that is within a function normally resides for the program to execute the function. Once the function's execution has been completed and the results has been returned to where the function is called, all data within the local scope will be deleted.*

8. In a procedural style, in any language you like, write a function called Average, which accepts an array of integers and returns the average of those integers. Do not use any libraries for calculating the average. You must demonstrate appropriate use of parameters, returning and assigning values, and use of a loop. Note — just write the function at this point, we'll *use* it in the next task. You shouldn't have a complete program or even code that outputs anything yet at the end of this question.

```
static float Average(int[] arrayInt)
{
    int numOfElements = arrayInt.Length;
    int i = 0;
    int total = 0;
    while(i < arrayInt.Length)
    {
        total = total + arrayInt[i];
        i++;
    }

    float average = total/numOfElements;
    return average;
}
```

9. In the same language, write the code you would need to call that function and print out the result.

```
int[] array = { 12, 14, 56, 23, 39 };
float result = Average(array);

static float Average(int[] arrayInt)
{
    int numOfElements = arrayInt.Length;
    int i = 0;
    int total = 0;
    while(i < arrayInt.Length)
    {
        total = total + arrayInt[i];
        i++;
    }

    float average = total/numOfElements;
    return average;
}

Console.WriteLine("The average is " + result);
```

10. To the code from 9, add code to print the message “Double digits” if the average is above or equal to 10. Otherwise, print the message “Single digits”. Provide a screenshot of your program running.

```
int[] array = { 12, 14, 56, 23, 39 };
float result = Average(array);

static float Average(int[] arrayInt)
{
    int numOfElements = arrayInt.Length;
    int i = 0;
    int total = 0;
    while(i < arrayInt.Length)
    {
        total = total + arrayInt[i];
        i++;
    }

    float average = total/numOfElements;
    return average;
}

Console.WriteLine("The average is " + result);

if (result >= 10)
{
    Console.WriteLine("Double Digits");
} else
{
    Console.WriteLine("Single Digit");
}
```