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: Show the path name of the current directory
    c. mkdir: Make a new directory/folder
    d. cat: Output the content of a file or concatenate files.
    e. ls: list all files and folders in the current 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	Address of a store
Integer	Number of members in a group
Float	The average point of a semester
Boolean	If a person is old enough to vote

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 complier "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 **initialising** a variable.

The difference between the two is: Declaring is setting up a variable with a data type, while initializing assigns an actual value that is appropriate for the date type.

```
int a; // Declaring
a = 10; // Initializing
```

2. 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 variable used in a function to accept input values.

```
static void Greet(string name)
{
    Console.WriteLine($"Hello, {name}!");
}

public static void Main()
{
    Greet("An"); // Output: Hello, An!
}
```

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" refers to the region within a program where a variable or function is accessible.

- Global Scope: Variables accessible from anywhere in the program.

```
public class Program
{
    public static int x = 10; // Global scope within the program

    public static void Main()
    {
        Console.WriteLine(x); // Outputs: 10
        Foo();
    }

    public static void Foo()
    {
        Console.WriteLine(x); // Accesses the global variable x
    }
}
```

- Local Scope: Variables accessible only within a specific function or block.

```
public class Program
{
    public static void Main()
    {
        int y = 5; // Local scope
        Console.WriteLine(y); // Outputs: 5
    }

    public static void AnotherMethod()
    {
        Console.WriteLine(y); // Error: y is not accessible here
    }
}
```

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

STEP 1:

```
def Average(arr):
    sum = 0.0
    count = 0.0

    for num in arr:
        sum += num
        count += 1

    if count == 0:
        return 0.0
    return sum / count
```

STEP 2:

```
def main():
    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] #(a)
    ave = Average(arr)  # (b)
    student_id = "104844794"
    print("Average: ", ave)  # (c)
    print(f"Minh An Nguyen - {student_id}")  # (d)

main()
```

Output:

```
Average:  -8.258333333333333
Minh An Nguyen - 104844794
```

STEP 3:
Example code output for this step:

```
(base) minhan6559@MacBook-Air-cua-minhan6559 1.1 % python 1.1.py
Average:  -8.258333333333333
Minh An Nguyen - 104844794
Single digit
Average value negative
Smaller than my last digit
```

Finally on a SINGLE screenshot of the program (main line and function) running with its outputs here: