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:
        - Choose the directory path within the path
    b. pwd:
        - Show the current directory location
    c. mkdir:
        - make new directory folder
    d. cat:
        - show file content
    e. ls:
        - show all directory within the path

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	A country’s capital name
Integer	A vehicle’s total wheels
Float	The average temperature of a city
Boolean	Whether a car's headlights are on

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`	3.1415	Float
`1 + 2 * 3 + 4`		10	Integer

`a and False`	`a = True`	Falae	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 just declares the type

of the variable while initializing assigns the variable with a value

such as string, integer, Boolean.

Paste your example code below:

Declaring the variable

Int x; // declaring the integer x;

Initialising a variable

Int x = 10; initializing the integer x with the value 10.

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 that used to pass the

value through the function or procedures.

Paste your example code below:

```
def greetings(name):

    print('Hello, {name}")

greetings("Alex");
```

Output:

```
Hello, Alex
```

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 variable or function is accessible or valid. It determines where a variable can be referenced or modified in a program.
Types of scope:

1. Global scope
2. Local scope
3. Function scope

Paste your example code below:

Global scope:

X = 10

Def function():

Y = 20

Print("this is from local scope

:{y}")

Print("this is form outer scope: {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:

STEP 2:

5. Using the same preferred language, write the main line calling code you wouldneed 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 sim
   ply 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: your should have made **no changes** to your function.

Paste all of your example code below:

Step 1:

```
def average(numbers):
    if not numbers: #if there is no numbers in the list
        return 0

    sum = 0
    count = 0

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

    return sum / count

print(average([1, 2, 3, 4, 5]))
```

Step 2:

```
def average(numbers):
    if not numbers: #if there is no numbers in the list
        return 0

    sum = 0
    count = 0

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

    return sum / count

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]
student_name = "Khant Thu Aung"
student_id = "105292912"

result = average(data_values)
print(f"Average: {result}")
print(f"Student Name: {student_name}")
```

```
print(f"Student ID: {student_id}")
```

Paste your example code's output here:

Step 1:
3.0
Step 2:
Average: -8.258333333333333
Student Name: Khant Thu Aung
Student ID: 105292912

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.

Paste your example code’s output here:
Average: -8.258333333333333
Student Name: Khant Thu Aung
Student ID: 105292912
Single digit
Average value negative
Larger than my last digit

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.

11P.py > ...

```python
def average(numbers):
    if not numbers: #if there is no numbers in the list
        return 0

    sum = 0
    count = 0

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

    return sum / count

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]
student_name = "Khant Thu Aung"
student_id = "105292912"

result = average(data_values)
print(f"Average: {result}")
print(f"Student Name: {student_name}")
print(f"Student ID: {student_id}")

if (result >= 10):
    print("Multiple digits ")
else:
    print("Single digit")

if result < 0:
    print("Average value negative")

#Extract the last digit
last_digit = int(str(result)[-1])
last_digit_id = int(str(student_id)[-1])

if last_digit > last_digit_id:
    print("Larger than my last digit")
elif last_digit < last_digit_id:
    print("Smaller than my last digit")
else:
    print("Equal to my last digit")

```