JAMES COOK UNIVERSITY AUSTRALIA

Student Number |__|__|__|__|__|__|__|__|

Family Name _____

First Name _____

CP1404 Programming II

Final Exam SP53, 2021
Online Examination for JCUS student only
College of Science and Engineering

Exam Duration

3 hours (this includes uploading of answers to the assigned submission folder within the given duration and **Late submissions will not be processed)**

Student to complete their examination at least 20 minutes before the due time and submit in LearnJCU at least 15 minutes before due time.

Technical Issue

Do a screenshot of the issues with the date & time on the computer and email to exams-singapore@jcu.edu.au with the following details:

1. Name, Student ID, and Subject Code
2. Details of the issue
3. Contact Number we can call you at

Exam Advisory

Students are advised not to share their answer scripts with other students, copy material from online and offline sources as well as discuss their answers.

Students caught through plagiarism checks, will be referred to the Exam misconduct panel with a possibility of their exam attempt being voided or shown a failure in the subject.

Exam Conditions:

This is an OPEN BOOK exam. You may use whatever resources you want, except other people. You should not ask questions of people via email, online forums, phone, chat, or any other means. You may ask questions about this exam via the Collaborate session during reading time or after reading time by sending an email to exams-singapore@jcu.edu.au

Note that any questions about content will likely not be answered. We will answer questions about the expectations or understanding of the questions, but not how to solve the problems you are expected to be able to solve.

You must do your own work. Submissions that are detected to be too similar to that of another student or to code found online (anything that is likely not produced by you) will be dealt with according to the College procedures for handling plagiarism and may result in serious penalties.

Your code will be evaluated in terms of best practice including identifier naming, choice of code constructs, and principles like DRY (Don't Repeat Yourself), SRP (Single Responsibility Principle), etc. The goal is not just to write answers that work, but to write good code based on what you have been taught in this subject.

The exam is worth 68 marks in total.

Please answer all questions in the provided Python files, as instructed in these requirements.

Some questions will request you to use a particular file and templates for these files have been provided that you should use. For the rest of the questions, please use the file called exam_submission.py.
You may use additional files if needed (and you can use temporary files while working on a single question), but you must ensure that each question's answer is clearly identifiable. Where you are asked to "explain" and write code, do this in the Python file with your explanations in comments.

You must not put your assignment work on GitHub or any other public place.

Submit a zip file containing only your code files for this exam assignment. Please name the file like: FirstnameLastnameExam.zip e.g. if your name were Miles Davis, the filename would be MilesDavisExam.zip. Submit your single zip file by uploading it on LearnJCU under Assessment (click on the title of this assessment).

1. **[3 marks]** Explain the meaning of the error message produced by the following code, specifically as it relates to this code. Suggest how you would overcome the error without using exceptions handling.
   The error message is:
   `TypeError: unsupported operand type(s) for +: 'MyValue' and 'MyValue'`

```
class MyValue:

    def __init__(self, value):
        self.value = value

value1 = MyValue(10)
value2 = MyValue(15)
print(value1 + value2)
```

2. **[2 marks]** Explain the meaning of the error message produced by the following code, specifically as it relates to this code.
   The error message is:
   `TypeError: 'str' object does not support item assignment`

```
text = "ogre"
text[1] = "G"
```

3. **[1 mark]** Write a good commit message for the following change in code

```
# Code before commit:
def main():
    height = float(input("Height: "))
    weight = float(input("Weight: "))
    bmi = weight / (height ** 2)
    print("Your BMI is", bmi)

main()

# Code after commit:
def main():
    height = float(input("Height: "))
    weight = float(input("Weight: "))
    bmi = calculate_bmi(height, weight)
    print("Your BMI is", bmi)

def calculate_bmi(height, weight):
    return weight / (height ** 2)

main()
```

4. **[2 marks]** Consider a class that represents an "electric vehicle".
   Write the most appropriate names for each of the following parts of that class:

   a. the module (file) name
   b. the class name
   c. a method that charges the electric vehicle
   d. an attribute variable that represents whether the electric vehicle is fully charged

5. **[5 marks]** For each of the scenarios below, explain what data structure (Python variable type) would be the best choice and why.

   a. Storing a user's weight
   b. Storing a year's worth of temperature reading
   c. Storing a collection of colours that allows the coder to get their colour codes based on their common names
   d. Storing the on/off state of a light bulb
   e. Storing a student's details (student number, name, address, etc.)

6. **[3 marks]** Given the below pairs of classes, describe their relationship using the following options:
   - is unrelated to
   - is an instance of
   - is a child of
   - is composed of

   e.g. if the pair is **"Animal - Dog"**, the answer is **"Dog is a child of Animal"**
   Note that you might need to swap the order to use the provided relationships.

   a. Person - BodyPart
   b. Avocado - Food
   c. Desk - Computer
   d. Microphone - InputDevice
   e. Student - Lecturer
   f. Jim - Person

7. **[3 marks]** The following code has room for improvement.
   Refactor the code and explain your reason(s).

```
# Refactor the code and explain your reason(s)
length = float(input("Length? "))
width = float(input("Width "))
if length * width >= 30:
    print("Room of size {} sqm => Big".format(length * width))
elif length * width >= 10:
    print("Medium room of size {} sqm".format(length * width))
else:
    print("Your room is small at only {} sqm".format(length *
width))
```

8. **[3 marks]** The following code does not work. You can read the programmer's intention in the comments below.
   Explain why it is not working.
   Then, rewrite it, fixed so that the intention is achieved.

```
# Start with lowercase state names
things = ["qld", "nsw", "act"]
print("Original:", things)

# Make them uppercase
for thing in things:
    thing = thing.upper()
print("Uppercase names now:", things)
```

9. **[4 marks]** Refactor the following code (rewrite it to work the same way but be better) to improve it based on the best-practices you have been taught, including the Single Responsibility Principle (SRP) for function design.
   Also, briefly explain how you have refactored it and why. How is the code better in your new version?

```
def main():
    in_file = open("data.txt")
    data = get_data(in_file)
    in_file.close()
    average(data, len(data))


def get_data(in_file):
    values = []
    for line in in_file:
        values.append(float(line))
    return values


def average(data, length):
    average = sum(data) / length
    print("The average is", average)


main()
```

10. **[4 marks]** Write an input-validation loop to get a weightlifting participant's weight in kilograms as a valid float, then print the weight in pounds (1 kg = 2.2 pounds). Your code should produce the output below (user input follows the "Weight (kg): " prompt):

```
Weight (kg): seventy
Invalid float
Weight (kg):
Invalid float
Weight (kg): 10
Weight must be between 61 and 109
Weight (kg): 110
Weight must be between 61 and 109
Weight (kg): 80
80.0 kg = 176 pounds
```

11. **[2 marks]** Complete the missing function body (replace only the ??) for the function below such that the following print statements produce the output specified in the comments. You may not change or add anything else in the code.

```
def function(??
    return x + y + z

print(function(2, y=5))  # prints 27
print(function(100))  # prints 125
```

12. **[2 marks]** Given the following dictionary definition, write code to produce the output below.
Notice that the output is sorted. Your code should work for any similar dictionary.

```
d = {4: "good", 1: "hello", 5: "subject", 2: "welcome", 3:
"exam"}
```

Output:

```
1 is h
2 is ww
3 is eee
4 is gggg
5 is sssss
```

13. **[3 marks]** Given the following data strings, write code of no more than 5 lines that extracts the uv value and stores it as a float in a variable named uv.
Do not use any additional libraries.
Note that your code should work for each of the following strings (test it with both), as well as any similar data strings.

```
data_string = "date='12/11/2020', temperature='23.1c',
uv='6.43'"
data_string = "uv='11.05', date='01/10/2019',
temperature='9.7c'"
```

14. **[3 marks]** Given the list of numbers below, write expressions to produce the output.
In each of the 6 expressions, you must use the list at least once.
You may NOT use the actual list element values in your expressions.

```
numbers = [10, 20, -13, -20, 55, 107, 200, -100, -222]
```

Output:

```
-222
[-13, -20, 55, 107, 200]
[10, -20, 200]
[-222, -100, 200, 107, 55]
[10, 20, 55, 107, 200]
There are 9 numbers
```

15. **[3 marks]** Given the variable definition below, write code to produce the provided output.

```
pairs = ((2010, 'cm'), (1234, 'm'), (129.59, 'km'), (41231, 'm'),
(67, 'km'))
```

Output:

```
    2010 cm   ->    0.02 kilometres (converted).
 1234.56 m    ->    1.23 kilometres (converted).
  129.59 km   ->  129.59 kilometres.
   41231 m    ->   41.23 kilometres (converted).
      67 km   ->   67.00 kilometres.
```

Notes:
- You can assume that the first part in each pair (distance) is always less than 8 digits.
- You can assume that the second part in each pair (distance in km) is always less than 10000.
- Notice that values with second part as ‘cm’ or ‘m’ has to be converted to km and indicated as "converted".

16. **[2 marks]** Given the variable definition below, write code to produce the provided output.

```
names = ["Jim Lee", "Sir Ali Baba", "Kathy", "Bob Tim"]
```

Output:

```
1. Jim Lee - 2 word(s)
2. Sir Ali Baba - 3 word(s)
3. Kathy - 1 word(s)
4. Bob Tim - 2 word(s)
```

17. **[2 marks]** Given the variable definition below containing a list of tuples where each tuple contains a string and number, write code to create a new variable called `new_values` that contains the provided output (as it would appear if printed) by using sorting and a list comprehension.
You are allowed to import additional libraries.
Produce your answer in no more than 4 lines.
Note that it should work for any similar values.

```
values = [('Bob', 4), ('Jo', 5), ('Harry', 3), ('Al', 3),
('Gwen', 3)]
```

Output (just print the new variable, no extra formatting):

```
['3-Gwen', '3-Harry', '4-Al', '4-Bob', '5-Jo']
```

18. **[2 marks]** The following code does not work. You can read the programmer's intention in the comments below.
Explain why it is not working.
Then, rewrite it, fixed so that the intention is achieved.

```
def init_this(things):
    """Initialize empty list"""
    things = [1, 2, 3]

some_things = []
init_this(some_things)
print(some_things)   # should be [1, 2, 3]
```

19. **[3 marks]** Given the variable definition below, write 3 list comprehensions to set the values as shown and explained below.

```
values = [4, 2, -3, 12, 8, 2, 9, -3]
```

unique_powers = the squares of the non-duplicate values in sorted order:
[4, 9, 16, 64, 81, 144]

products = the values (in original order) as the product of each value and its respective index:
[0, 2, -6, 36, 32, 10, 54, -21]

bigger_numbers = a list of the values that are bigger than the average value in the list:
[4, 12, 8, 9]

Your code should work for any similar list of numbers.
Each of your 3 answers should be a single line and include a list comprehension.

20. **[5 marks]** Given the provided file, "original.txt", write code in **musicians.py** to produce a new file,
"musicians.txt" that looks like "sample_musicians.txt".
Notice:
- The header line is the same in the original as the output file
- The lines are sorted by the musicians' first names
- The birth dates have been rearranged to be year-month-day (instead of day/month/year)
- The formatting of each line is different from the original

As usual, your code should work for any similar file (e.g. different header, different musician details, more or fewer number of lines, etc.).
Hint: As always, you would benefit from approaching this question incrementally. A good middle goal would be able to produce and print a list of stored data like:

```
[['Ella Fitzgerald', ('25', '4', '1917')], ['Harry Connick
Jr.', ('11', '9', '1967')], ['Louis Armstrong', ('4', '8',
'1901')], ['Michael Buble', ('9', '9', '1976')], ['Nina
Simone', ('21', '2', '1933')], ['Ray Charles', ('23', '9',
'1930')]]
```

21. **[3 marks]** Write the missing class, Thing in **thing.py**.
See the code in things.py and the output it produces when run, shown below.
Your job is to figure out from the output and the code in things.py how to write the Thing class
so that it produces the same output when you run it. Do not change things.py.

Output:

```
0. I'm Paul (0).
0. I'm pAUL (0).
...
1. I'm Simon (0).
1. I'm sIMON (5).
...
2. I'm Art (0).
2. I'm aRT (6).
...
3. I'm Garfunkel (7).
3. I'm gARFUNKEL (34).
...
```

22. **[8 marks]** Write the missing client code program, **stamps.py**.
Given the class Stamp (stamp.py) and the data file stamp_data.txt, write code for a main program in stamps.py that produces the output below. Do not change stamp.py
Note: As always, your code should work properly for a different data file.
The random stamp is randomly selected.
The countries are in ascending order.
If the rarity of the random stamp is True, (Rare) will be printed, otherwise (Common) is printed.

Output:

```
You have 16 stamps (7 rare).
Stamps with most number of denominations are from Australia
You have stamps from these 5 countries:
Australia, England, Hong Kong, New Zealand, South Africa
Today's random stamp is:
Hong Kong $2.60 (Common)
```

- End of Paper -