CSC108 FSG Midterm 2

Question 1: What will the following code output?

```
my_list = [1, 2, 3, 4, 5]
sliced = my_list[::-2]
print(sliced)
```

a) `[5, 3, 1]`

b) `[1, 3, 5]`

c) `[5, 4, 3, 2, 1]`

d) `[5, 3]`

Question 2: Consider the following function

```
def read_file():
    file = open("example.txt", "r")
    content = file.read()
    return content
```

Will this function always work without errors? Why or why not?

Question 3: How do you access the value of "history"?

```
dict = {
    "class" : {
        "student" : {
            "name" : "Mike" ,
            "marks" : {
                "physics" : 80,
                "history" : 70}
        }
    }
}
```

a) `dict[0][0]['marks']['history']`

b) `dict['class']['student']['marks']['history']`

c) `dict['class']['student']['history']`

d) `dict['class']['student'][0]['history']`

Question 4: What will the regex pattern ^[a-zA-Z]+$ match?

a) `A string containing only letters.`

b) `A string containing letters and numbers.`

c) `A string containing at least one letter.`

d) `A string containing letters and spaces.`

Question 5: Which of the following regular expression patterns will match an email address in the format utorid@mail.utoronto.ca, Where the **utorid** consists of lowercase letters and numbers and starts with a letter?

a) `^[a-zA-Z0-9]+@mail\.utoronto\.ca$`

b) `^[a-z0-9]+@mail.utoronto.ca$`

c) `^[a-z0-9]+@mail\.utoronto\.ca$`

d) `^[a-z]+@mail.utoronto.ca$`

e) `^[a-z][a-z0-9]+@mail\.utoronto\.ca$`

Question 6: What will the following code output?

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
my_dict['d'] = my_dict['b'] + my_dict['c']
my_dict['a'] = my_dict['d'] - my_dict['a']
print(my_dict)
```

a) `{'b':2, 'c':3, 'd':5}`

b) `{'a':1, 'b':2, 'c':3, 'd':5}`

c) `{'b':2, 'c':3, 'a':4, 'd':5}`

d) `{'b':2, 'c':3, 'd':6}`

Question 7: Select all statements that are true about the data types in Python:

a) List is mutable, meaning its elements can be changed after it is created.

b) Dictionary is mutable, meaning it can be altered after it is defined.

c) Dictionary keys can be any data type, such as lists and tuples.

d) Dictionary can be accessed by its index like Lists.

e) Tuple datatype is mutable, meaning its elements can be updated after it is created.

f) Any kind of Tuples can be used as a key for Dictionary datatype.

Question 8: Complete the code below

```python
def max_nested_sum(lst: list[list[int]]) -> list[int]:
    """
    Return the sublist with the maximum sum of elements from the
    nested list. If there are multiple sublists with the same
    maximum sum, return the first occurence.


    >>> max_nested_sum([[1, 5, 3], [4, 6, 2, 7], [8, 3, 9, 10]])
    [8, 3, 9, 10]
    >>> max_nested_sum([[1, 2], [2, 3], [5]])
    [2, 3]
    >>> max_nested_sum([[1, 1, 1], [2, 2, 2, 2], [3, 3, 3]])
    [3, 3, 3]
    >>> max_nested_sum([[-1], [], [2, -5]])
    []
    """
    # TO-DO
```

Question 9: Complete the code below

```python
def most_frequent_fruit(data: list[tuple[int, str]]) -> str:
    """
    Return the name of the fruit that appears most frequently
    (i.e. the one with the highest total quantity). You can
    assume no ties will occur(unique maximum will always exist).

    >>> most_frequent_fruit([(5, "apple"), (3, "banana"),
    (8, "cherry"), (6, "banana")])
    "banana"

    >>> most_frequent_fruit([(10, "apple"), (20, "banana"),
    (10, "cherry"), (15, "apple")])
    "apple"
    """
    # TO-DO
```

Question 10: You are tasked with writing a function to analyze a log file. Each line in the log contains a timestamp, a log level (e.g., "INFO", "ERROR", "DEBUG"), and a message. The function should:

- Count how many times each log level appears in the file.
- Return the counts in a dictionary where the keys are log levels (e.g., "INFO", "ERROR", "DEBUG") and the values are the counts of those log levels.

Function Signature:

```
def count_log_levels(file_path: str) -> dict[str, int]:
```

Input: `file_path` (str): The path to the log file.
Output: A dictionary where the keys are log levels, and the values are the counts of each log level.
Example: Given the following content in the log file `log.txt`:

```
2024-11-01 10:00:00 INFO Start of process
2024-11-01 10:05:00 ERROR Something went wrong
2024-11-01 10:10:00 INFO Process continues
2024-11-01 10:15:00 DEBUG Debugging the issue
2024-11-01 10:20:00 ERROR Another error occurred
2024-11-01 10:25:00 INFO Process finished
```

Calling `count_log_levels("log.txt")` should return the following dictionary:

```
{"INFO": 3, "ERROR": 2,"DEBUG": 1}
```