Name: ______________________________

Email Address (UW Net ID): ______________________ @uw.edu

Section: ______________________________

CSE 160 Summer 2023 - Final Exam

Instructions:

- You have **60 minutes** to complete this exam.
- The exam is **closed book**, including no calculators, computers, phones, watches or other electronics.
- You are allowed a single sheet of notes for yourself.
- We also provide a syntax reference sheet.
- Turn in *all sheets* of this exam, together and in the same order when you are finished.
- When time has been called, you must put down your pencil and stop writing.
    - **Points will be deducted if you are still writing after time has been called.**
- You may only use parts and features of Python that have been covered in class.
- All questions assume Python version 3.7, as we have been using all quarter.
- You may ask questions by raising your hand, and a TA will come over to you.
- If we discover typos or useful clarifications during the exam, we will add them to an Errata document presented on the screens.

Good luck!

Question	**Points**
Question 1	7
Question 2	5
Question 3	4
Question 4	5
Question 5	5
TOTAL	**27**

Question 1 [7 Pts]:

Write a function `display_recipies` for the `RecipeCollection` class that prints the phrase “`Recipe Collection`” and then the name and ingredients of the first three recipes in self.recipe. If there are **less than** three recipes, the function should print out the name and ingredients of all the recipes. If there are no recipes, the function should not print anything besides the line `'Recipe Collection”`.

```
class Recipe:
    def __init__(self, name, ingredients):
        self.name = name
        self.ingredients = ingredients

    def display_info(self):
        print("Name:", self.name)
        print("Ingredients:", self.ingredients)

class RecipeCollection:
    def __init__(self):
        self.recipes = []

    def add_recipe(self, recipe):
        self.recipes.append(recipe)

    def display_recipes(self):
        print("Recipe Collection:")
        if len(self.recipes) > 3:
            for i in range(3):
                self.recipes[i].display_info()
        else:
            for recipe in self.recipes:
                recipe.display_info()
```

Question 1 (continued)

For example, the following client code will have the following output

```
collection = RecipeCollection()
collection.display_recipes()
recipe1 = Recipe("Chocolate Cookies", ["flour", "sugar", "chocolate"])
collection.add_recipe(recipe1)
recipe2 = Recipe("Strawberry Shortcake", ["strawberry", "sugar",
"flour"])
collections.add_recipe(recipe2)
collection.display_recipes()
```

Expected output

```
Recipe Collection:
Recipe Collection:
Name: Chocolate Cookies
Ingredients: ['flour', 'sugar', 'chocolate']
Name: Strawberry Shortcake
Ingredients: ['strawberry', 'sugar', 'flour']
```

Question 2 [5 Pts]:

What is the output of the following code, which passes several variables into a function foo()?

Note: There are 5 print statements

```
def foo(list1, list2, my_string):
     list1.append(4)
     my_string += "aaa"

     list1 = list2
     list2 = [5, 6, 7]

     list1.insert(0, 2.2)
     list2.pop()
     print(list1)
     print(list2)
     return my_string[8]


list1 = [1, 2, 3]
list2 = [8, 9]
my_string = "string!"
my_string = foo(list1, list2, my_string)
print(list1)
print(list2)
print(my_string)
```

```
[2.2, 8, 9]
[5, 6]
[1, 2, 3, 4]
[2.2, 8, 9]
a
```

Question 3 [4 Pts]:

A student has written the function check_sorted(nums), which checks if a given list is sorted. The function returns True if the list is sorted, and False otherwise. The input is guaranteed to have at least one element.

The student also made some style issues. Please note the four main style issues that the student should fix, and the corresponding change to fix the issue.

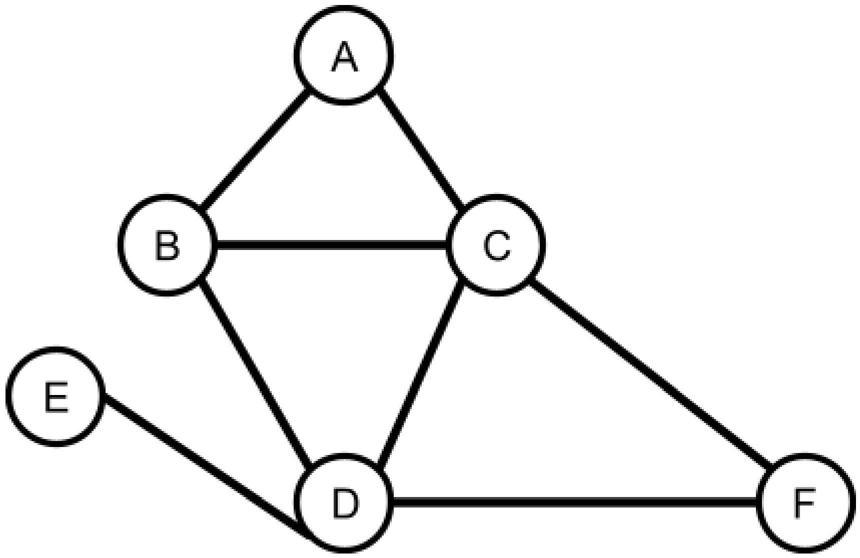
```
def check_sorted(nums):
    prev_num = nums[0]
    is_sorted = True
    for my_list in range(len(nums)):
        if prev_num>nums[my_list]:
           is_sorted = False
    if is_sorted:
        return True
    else:
        return False
```

Style issue	Fix
Loop variable naming	Change my_list to something more descriptive like index
Space between “>” operator	Prev_num > nums[my_list]
Bad boolean zen	Return is_sorted instead of using if statements
Iterates through range(len(nums)) instead of just nums	Iterate through num in nums instead of indices
No docstring	“”” Param: list of numbers. Function checks if the list is sorted Returns true if the given list is sorted, false otherwise ”’”
Prev_num not updated	Update prev_num = nums[my_list]

Question 4 [5 Pts]:

Write a function `is_connected(graph, vertex1, vertex2)` that takes in a graph and two strings representing vertices as input. The function should return True if `vertex1` and `vertex2` are connected directly (ex: share an edge), and False otherwise.

For example, given the following graph, `my_graph`, the following function calls would produce the following output



```
is_connected(my_graph, A, C) # Returns true since A and C are directly connected
is_connected(my_graph, A, D)  # Returns false since A and D are not directly connected
```

```
def is_connected(graph, vertex1, vertex2):
     return vertex2 in graph.neighbors(vertex1)
```

Question 5 [5 Pts]:

Please write the output of this code.

```
a = {1, 3, 5, 1, 5, 9}
b = [5, 1, 4, 2, 3, 1, 2, 1]

b_new = dict()
for num in b:
    if num in a:
        if num not in b_new:
            b_new[num]=0
        b_new[num] += 1

c = a | set(b)
d = a & set(b_new.values())
e = [num ** 2 for num in a]
f = [c for c in b if c not in a]

print(b_new)
print(c)
print(d)
print(e)
print(f)
```

```
{5: 1, 1: 3, 3: 1}
{1, 2, 3, 4, 5, 9}
{1, 3}
[1, 9, 25, 81]
[4, 2, 2]
```