CSE 8A/6R Fall 2025 Homework 2

Due: Thursday, October 16, 2025 before 11:59pm

Full Name: _____ PID: _____

Homework 2 Topics Covered:

1. Conditional statements
2. Lists: creating, indexing, slicing, list membership, mutation
3. Strings

5 core competencies:
1. Code Tracing, 2. Code Comprehension, 3. Code Writing, 4. Debugging, and 5. Testing

Submission

Questions 1 - 4:

- **Handwritten** in the given boxes only.
- Scanned/uploaded to Gradescope **Homework 2 Questions 1 - 4** assignment.

Question 5 (process journal):

- The **process journal** can be either handwritten or typed into a document, submitted as a PDF.
- Uploaded to Gradescope **Homework 2 Question 5 Process Journal** assignment.

Point breakdown for Homework 2:

- **Exemplary (3 homework points):** All problems receive a "Pass."
- **Satisfactory (2 homework points):** At least 4 out of 5 questions receive a "Pass."
- **Needs Improvement (1 homework point):** At least 1 out of 5 questions receive a "Pass."
- **Incomplete (0 homework points):** No problems receive a "Pass," or the assignment was not submitted, or the assignment is ungradable (e.g., illegible handwriting), or if the submission does not follow the instructions (e.g., homework is typed instead of handwritten)

Your work will be marked **Ungradable** if your submission cannot be assessed because of issues that prevent interpretation of the work. This may include, but is not limited to, **illegible handwriting**, a corrupted or unreadable file, missing pages or sections, or work that is completely off-topic or irrelevant to the assignment.

1. [Code Tracing] Trace the following code and answer the questions.

```
nums = [2, 4, 8, 16, 32]
nums[2] = 99
nums.append(64)
print(nums)
print(nums[1:4])
```

A) What is the output (i.e., what will be printed) after executing the lines above? Your work should show any changes to `nums` and mark the index of each item in the list.

Show your work:

Output of `print(nums)`:

Output of `print(nums[1:4])`:

B) What happens if we add and execute the following line to the end of the code snippet above?

```
print(nums[len(nums)])
```

Answer:

Explanation:

2. [Code Tracing] Examine the following code and answer the questions that follow.

The following exercise asks you to trace the execution path of a sequence of `if/else` statements. Remember that an `if` statement only executes its body (e.g., a `print()` function) if its condition is `True`.

```
if <condition_A>:
   print("A")
if <condition_B>:
   print("B")
if <condition_C>:
   print("C")
else:
   print("D")
```

A) Is it possible for the code to print both "A" and "D"?

Answer:

Explanation:

B) Is it possible for the code to print all four letters: "A", "B", "C", and "D"?

Answer:

Explanation:

3. [Code Comprehension, Testing] In this question, your goal is to understand what the mystery function does.

```
def mystery(some_list):
    if len(some_list) >= 2:
        temp = some_list[0]
        some_list[0] = some_list[len(some_list) - 1]
        some_list[len(some_list) - 1] = temp
    return some_list
```

A) Below are four tests of the `mystery` function.
For each test, **predict the output.**

Test #1: `print(mystery([10, 20, 30]))`

Output #1:

Test #2: `print(mystery([42]))`

Output #2:

Test #3: `print(mystery(["8A", "6R"]))`

Output #3:

Test #4: `print(mystery([]))`

Output #4:

B) Explain in plain English the purpose of the mystery function.

4. [Debugging]
A shipping depot uses the Python function `calculate_container_category(weight)` to classify containers based on their weight in kilograms (kg). The code runs without error, but fails the test cases.

```
# This function calculates the category of a container based on its weight
# in kilograms (kg). The categories are defined by the required logic below:
# Light: weight < 3 kg
# Medium: 3 kg <= weight <= 10 kg
# Heavy: weight > 10 kg
# Error: weight <= 0 kg (This should catch non-positive weights)

def calculate_container_category(weight):
    category = None
    if weight > 10:
        category = "Heavy"
    elif weight <= 10:
        category = "Medium"
    elif weight < 3:
        category = "Light"
    else:
        category = "Error"
    return category

# Test Case 1: print(calculate_container_category(2))
# Expected output: "Light"
# Actual output from buggy code: "Medium"

# Test Case 2: print(calculate_container_category(-1))
# Expected output: "Error"
# Actual output from buggy code: "Medium"
```

Part 1: Identify and Explain the Logical Bugs
Analyze the `if/elif/else` structure in the provided Python function. Explain the primary logical flaw that causes the incorrect classification in both test cases.

A) Which line contains the conditional statement that incorrectly catches the input of 2 kg?

B) Explain: Why does the structure of the `if/elif` ladder prevent the correct `Light` category check from ever being reached for weights like 2 kg or 1 kg?

C) Explain: Why will the required Error category (for `weight <= 0`) never be executed in the current buggy code, even though it is present in the intended logic?

Part 2: Fix the Code

D) Rewrite the entire function below. Your corrected code must be logically sound and correctly ordered to pass all stated classification rules and test cases.

```
def calculate_container_category(weight):
    # Your corrected if-elif-else structure goes here









    return category
```

5. [Code Writing, Testing] Automating the calculation of Social Learning Credit in CSE 8A-6R!

For this question, you are submitting a ***process journal***. For detailed instructions, please check the instruction document: https://tinyurl.com/ProgrammingProcessJournal. Be sure to read this before you start implementing!

In our syllabus, we described how to calculate the social learning grade for the week:

Each week, there are **5 events**: 1 reading quiz, 2 lectures, 1 lab, and 1 study group.

Your weekly social learning score **for each week** is determined as follows:

- **Exemplary (3 social points):** You receive a "Pass" for at least 4 events, including a "Pass" on **both** the weekly lab **and** the reading quiz.
- **Satisfactory (2 social points):** You receive a "Pass" for at least 3 events, including a "Pass" on **at least one** of the weekly lab **or** the reading quiz.
- **Needs Improvement (1 social point):** You receive a "Pass" for at least 2 events.
- **Incomplete (0 social points):** You receive a "Pass" for fewer than 2 events.

Now that you have learned programming, let's automate the weekly social point calculation using Python knowledge! The properties of the function are provided for you:

- Name of the function: `social_learning_grade`
- Parameters: all parameters of this function are integers (ints).
    - `reading_quiz`: `1` if you completed the Cogniterra reading quiz, `0` if not.
    - `lectures`: 0 to 2, a count of how many lectures you attended.
    - `lab`: `1` if you attended the lab, 0 if not.
    - `study_group`: `1` if you attended the study group, `0` if not.
- Data type of return value: integer (int), representing the total social points you earned for that week.

Example of calling and testing the function:

```
print(social_learning_grade(1, 2, 0, 1)) # Expected output: 2
```