# CSE 8A/6R Fall 2025 Homework 5

**Due:** Thursday, November 20, 2025 before 11:59 pm

Full Name: _____ PID: _____

**Homework 5 Topics Covered:**
1. Nested for loops and 2D lists 2. Debugging

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

# Submission

**Questions 1 - 5:**
- **Must be handwritten** in the given boxes only.
- Scanned/uploaded to Gradescope **Homework 5 Questions 1 - 5** assignment.
- *If any of the questions 1 - 5 are typed*, *then your work will be considered* ***ungradable****.*

**Question 6 (process journal):**
- The **process journal** can be either handwritten or **typed (preferred)** into a document.
- **In addition, you'll need to submit the Python file (.py) to the problem. See submission instructions below.**
- Uploaded to Gradescope **Homework 5 Question 6 Process Journal** assignment.

When you submit your work, **please make sure to assign the correct pages to the questions**. Otherwise, your work will be considered **ungradable**.

**Point breakdown for Homework 5:**
- **Exemplary (3 homework points):** All problems receive a "Pass."
- **Satisfactory (2 homework points):** At least 4 out of 6 questions receive a "Pass."
- **Needs Improvement (1 homework point):** At least 1 out of 6 questions receives 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, not answering in the boxes provided, not assigning pages to the question).

1. [Code Writing, Debugging] The Python code below is intended to *identify and print any digits* found within the **`sentence`** string. However, it contains **three errors**, each isolated to a single line, and can be fixed by modifying **only** those lines.

Code:

```
1 sentence = B3li3veInM1racl3s
2 digits = "1234567890"
3 for digit in digits
4    if digit in sentence:
5    print("The sentence contains digit", digit)
6 # Expected Output:
7 # The sentence contains digit 1
8 # The sentence contains digit 3
```

A) Identify the line where the errors occur and explain how to fix it. You don't need to show the errors in any specific order.

Error # 1: Line _____

Explain how to fix, or write out the fixed line below:

Error # 2: Line _____

Explain how to fix, or write out the fixed line below:

Error # 3: Line _____

Explain how to fix, or write out the fixed line below:

2. [Code Tracing, Debugging] Observe the code below and answer the questions below.

Code:

```
1  def has_consecutive_duplicates(nums):
2      if len(nums) < 2:
3          return False
4      for i in range(len(nums)):
5          if nums[i] == nums[i + 1]:
6              return True
7      return False
8
9  print(has_consecutive_duplicates([1, 2, 3, 4, 5]))
10 # Expected Output: False
```

Output:

```
IndexError: list index out of range
```

A) Show how the variable `i` and values `nums[i]` and `nums[i + 1]` change with each iteration of the `for` loop. Briefly explain or highlight how the `IndexError` occurs. You don't have to draw stack frames.

B) Would an error occur if we *replace* line 9 with the following test?

```
print(has_consecutive_duplicates([1, 2, 2, 3, 4])) # Expected: True
```

Answer to the question (circle one): Yes, an error will occur | No, an error will not occur

Explain your answer:

3. [Code Writing, Debugging] Hailey wrote the Python function `add_by_n()` to **mutate** (i.e., change in place) a list by adding a value n to every element. Their implementation fails to produce the expected output. Your task is to analyze the provided code, identify the *logical error*, and use **memory diagrams** to explain why the code fails to update the list.

| | Code: | Output: |
|---|---|---|
| 1 | `def add_by_n(my_list, n):` | |
| 2 | `  for item in my_list:` | [1.5, 2.8, 3.0] |
| 3 | `      item += n` | |
| 4 | | |
| 5 | `input_list = [1.5, 2.8, 3.0]` | |
| 6 | `add_by_n(input_list, 5)` | |
| 7 | `print(input_list) # Expected: [6.5, 7.8, 8.0]` | |

**Part 1: Identify and Explain the Logical Bug**

A) Identify the **line** containing the logical bug. Then, use **memory diagrams** (showing the call stack, stack frames, references, and objects in memory) to trace the execution and **show precisely why the actual output is produced**. Use diagrams and text to trace the execution of the function call `add_by_n(input_list, 5)` to explain why `input_list` remains unchanged.

B) Explain: why is the actual output different from the expected output?

**Part 2: Fix the Code**

C) Rewrite the entire function below. Your corrected code must be logically sound and should show the expected behaviors.

```
def add_by_n(my_list, n):
   # Your corrected function definition (i.e., implementation goes here)
```

4. [Code Tracing] The following Python code manipulates a nested list called `seat_assignment`, which represents a classroom seating chart. The **outer list** represents the rows, and the **inner lists** contain the seat assignments (names or special assignments like "stairs"). Your task is to trace the execution of the code and show the final state of the seating chart after all lines have executed.

Code:

```
1  seat_assignment = [
       ["Sang",   "Fatimah", "stairs", "Michelle", "Shravi", "Masha"],
       ["Andrew", "Harsha",  "stairs", "D'ante",   "Mia",    "Anisha"],
       ["Prud",   "Joshua",  "stairs", "Steven",   "Renita"]
   ]

2  new_students = ["Liam", "Kyle", "stairs", "Ryan", "Kyla", "Janoj"]
3  seat_assignment.append(new_students)
4
5  seat_assignment[2].insert(4, "Kenneth")
6
7  seat_assignment[0][2] = "accessible seat"
```

A) The chart below visualizes the initial state of the `seat_assignment` list (after line 1 is executed). Complete the chart by tracing the effects of lines 2 to 7 to show the final seating arrangement after lines 2 - 7 are executed.

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | Sang | Fatimah | stairs | Michelle | Shravi | Masha |
| 1 | Andrew | Harsha | stairs | D'ante | Mia | Anisha |
| 2 | Prud | Joshua | stairs | Steven | Renita | |
| 3 | | | | | | |

B) What is the result of each print statement when executed immediately following the seating chart modifications? Assume that the following lines are written after line 7 in the code above.

Print # 1: `print(seat_assignment[3])`

Output:

Print # 2: `print(seat_assignment[1][4])`

Output:

Print # 3: `print(seat_assignment[0][5][3])`

Output:

5. [Code Comprehension] Amy is organizing ice cream orders for herself and two friends. She has recorded the ingredients in a 2D list called `orders`, where each row represents a type of ingredient (e.g., flavor, topping, fruit) and each column represents a different order (one person's full order).

Code:

```
1  orders = [
2     ["Vanilla",    "Chocolate", "Rocky Road"],
3     ["Sprinkles",  "Oreo",      "Nuts"],
4     ["Blueberry",  "Cherry",    "Banana"]
5  ]
6  size = len(orders)
7  new_orders = []
8  for i in range(size):
9     new_orders.append([])
10    for j in range(size):
11        item = orders[j][i]
12        new_orders[i].append(item)
13 print(new_orders)
```

A) **Explain in plain English** how the list generated by the code, `new_orders`, represents the data from the original `orders` list. Describe the general pattern or relationship between the two lists, without just listing the elements.

B) Based on the structure of the loops in the code, what assumption does the function make about the dimensions (i.e., number of rows and number of columns) of the 2D list (`orders`)? Use the following scenario to illustrate your answer: What would happen if the number of rows was different from the number of columns (e.g., 4 rows of ingredients for 3 people)?

6. [Code Writing, Testing] Average Rainfall (again)!

For this question, you are submitting a ***process journal***. For detailed instructions, please check the instruction document: https://tinyurl.com/ProgrammingProcessJournal. IMPORTANT: For this question, please submit your python file (.py) (in addition to your process journal) that you used to implement the program. Submission instructions are now available in the sample process journal document linked above.

**Question Description:**

The city of Nas Ogeid keeps track of annual rainfall in a 2D list. Design a program to calculate the average rainfall for each month, returned as a list.

**Name of the function:** `average_rainfall`

**Parameter:** `rainfall_data`, a 2D list. The outer list represents the months. The inner list contains *positive float values* (if any) representing the precipitation recorded on rainy days for that month.

**Data type of return value:** A *list of floats* where each element is the average rainfall for the corresponding month.

This function **must** use a nested for loop. As a *bonus (not required)*, you can think about how to use a helper function to break the problem down into smaller pieces!

**Sample test:**

```
rainfall_sample = [
   [], # no rain in September
   [3.2, 3.6, 2.2], # Three rainfall measurements in October
   [4.1, 4.6],  # Two rainfall measurements in November
]
print(average_rainfall(rainfall_sample)) # output should be: [0, 3.0, 4.35]
# Note: 3.0 is the average of [3.2, 3.6, 2.2]
# Note: 4.35 is the average of [4.1, 4.6]
```

**Special Case: Handling Zero Rainfall**
Pay careful attention to months where no rain was recorded (represented by an empty inner list, e.g., `[]`). The average rainfall for such a month must be 0 (zero), *not an error*.