Homework 5: Lists and Tuples in Python

Objective

This homework will help you practice **theory** and **practical coding** skills related to **lists** and **tuples** in Python. You'll answer conceptual questions, perform hands-on operations, and compare these two data structures in a short essay.

Part 1: Theory Questions

Terminology

List
Tuple
Indexing
Mutability
Immutability

Tip: Think about **how** each term relates to Python's approach to storing and organizing data. You can refer to class notes or official Python documentation for clarity.

Difference Between Lists and Tuples

Explain, in **your own words**, how lists differ from tuples in **structure**, **usage**, and **limitations**.

Indexing in Lists and Tuples

Describe how you **access** elements using indexing (both **positive** and **negative** indices).
Provide **brief examples** to illustrate indexing in **both** lists and tuples.

Part 2: Coding Problems

Problem 1: List Manipulation

Task: Write a Python program that demonstrates:

- **Creating** a list of integers.
- **Appending** new elements.
- **Removing** elements.
- **Accessing and modifying** elements at specific indices.
- **Printing** the final list.

Tip: Remember the common list methods like `append()`, `remove()`, and how to use indexing (e.g., `my_list[0]`) to read or modify an element.

Problem 2: Tuple Operations

Task: Write a Python program that shows how tuples work by:

- **Creating** a tuple of strings.
- **Accessing** elements using indexing.
- **Attempting** to modify an element (which should cause an error).
- **Concatenating** two tuples.
- **Printing** the final tuples.

Tip: Because tuples are **immutable**, you'll see a specific type of error when you try to change an element. Think about how to handle or demonstrate that in your code.

Problem 3: List vs. Tuple

Task: Write a **short essay** comparing and contrasting lists and tuples.

- Discuss **similarities** and **differences**.
- Provide **examples** of when you might use a list vs. a tuple.

Tip: Consider topics like **mutability**, **performance**, **common use cases**, and the **advantages** of each.

Submission Guidelines

Theory Answers:

Write your responses to the theory questions in clear, concise paragraphs or bullet points.

Coding Problems:

Submit your Python scripts (`.py` files). Each problem can be in a separate file or combined into one file with clear section headers.

Formatting:

Ensure your code follows **proper Python syntax** and **indentation**.

Include **comments** to explain key logic or clarify your approach.

References:

Feel free to use class notes, official Python documentation, or reputable online resources to support your answers.

Due Date:

Submit according to your instructor's specified deadline or platform.

Good luck, and happy coding!

Use this homework to reinforce your understanding of **lists** and **tuples**. Don't hesitate to experiment, practice different methods, and explore edge cases (like empty lists or single-element tuples) to deepen your learning.