Lab 1: Student Handout

Course: CS230-Data Structures, Instructor: Muhammad Kazim, Ph.D.

Slack Invitation:

https://join.slack.com/t/cs230datastructure/shared_invite/zt-3dsjv98bl-Bgofs55~_4ejOKM2NU_~yg

Google Classroom Invite:

https://classroom.google.com/c/ODA3MDU5NzI5MDE3?cjc=7jrmc4pu

Lab Week: 1

Topics: Algorithm Notation, Data Structure Operations, Complexity Analysis

Objective: To practice designing algorithms and analyzing their efficiency without writing code.

Introduction

Welcome to your first Data Structures lab! Today, we move from theory to practice. You will work on expressing algorithms clearly and judging their quality. The goal is to strengthen your foundational understanding before we start coding.

Activity 1: Algorithm Design (35 mins)

Goal: Translate simple problems into algorithms using pseudocode and flowcharts.

Instructions: In your groups, choose one of the problems below. Design a solution by:
- Writing clear, step-by-step pseudocode.
- Drawing a flowchart that represents the same logic.

Problems (Choose One):
A. The Even Counter

B. The Reverser
C. The Palindrome Checker
D. The 2D Sum

Use the space below for your work.

Pseudocode:

Flowchart: (Draw your flowchart with clear symbols: Oval=Start/End, Rectangle=Process, Diamond=Decision, Parallelogram=Input/Output)

Activity 2: Operation Identification (35 mins)

Goal: To recognize fundamental Data Structure Operations within existing algorithms.

Instructions: For each algorithm below, analyze each step and label which data structure operation is being performed. Choose from: **Traversal**, **Insertion**, **Deletion**, **Searching**, **Sorting**.

Algorithm 1: The Unique Filter

```
# Given a list 'data'
unique_list = []            # Operation: _____

for item in data:           # Operation: _____

    if item not in unique_list:  # Operation: _____

        unique_list.append(item) # Operation: _____
```

Algorithm 2: The Sorted Inserter

```
# Given a sorted list 'scores' and a new score 'n'

for index in range(len(scores)): # Operation: _____
    if n > scores[index]:
        continue
    else:
        scores.insert(index, n)  # Operation: _____
        break
```

Activity 3: Complexity Analysis (50 mins)

Goal: To apply Big O notation to classify algorithms based on their efficiency.

Instructions: For each algorithm description below, determine its most likely Time Complexity and provide a brief reason for your choice. Choose from: **O(1), O(log n), O(n), O(n log n), $O(n^2)$, $O(2^n)$.**

Algorithm Description	Big O Classification	Reason
1. Accessing the 5th element of an array.		
2. Printing every element in a linked list.		
3. Using nested loops to find all duplicate values in a list.		
4. Finding a number in a sorted list using Binary Search.		
5. Generating every possible subset of a set of items.		