DS2030 Data Structures and Algorithms for Data Science
Lab 6

September 23rd, 2025

Lab Instructions

- Create a folder named **"DS2030_<RollNo.>"** (all letters in capital) in **"home"** directory. Eg- **DS2030_142402022**
- Name the script files in the given format **"<your_roll_no>_<Name>_Lab6.py"**
- Make sure the **folder, files, classes, functions and attributes** are named as instructed in the lab sheet.
- We will not be able to evaluate your work if the folder is not properly named or is not located in the home directory.
- Make sure to save your progress before leaving the lab.
- Do not shut down the system after completing the lab.
- You are not allowed to share code with your classmates nor allowed to use code from other sources.

Hospital Patient Management System using AVL Trees

Design a Hospital Patient Management System for IIT Palakkad health center using an **AVL Tree**.

Attributes

Each **Patient Node** stores:

- `pid` : Unique Patient ID (integer).
- `name` : Patient's name.
- `age` : Patient's age (integer).
- `gender` : Gender (string).
- `disease` : Disease diagnosed (string).
- `doctor` : Doctor assigned (string).
- `admission_date` : Date of admission (string).
- `height` : Height of node for AVL balancing.

Starter Code

```
class Patient:
    def __init__(self, pid, name, age, gender, disease, doctor, admission_date):
        self.pid = pid
        self.name = name
        self.age = age
        self.gender = gender
        self.disease = disease
        self.doctor = doctor
        self.admission_date = admission_date
        self.height = 1  # Needed for AVL balancing


class AVLNode:
    def __init__(self, patient):
        self.patient = patient
        self.left = None
        self.right = None
        self.height = 1


class HospitalAVL:
    def __init__(self):
        self.root = None

    # AVL Utility Functions
    def get_height(self, node):
        pass

    def get_balance(self, node):
        pass

    def rotate_left(self, z):
        pass

    def rotate_right(self, z):
        pass

    # CRUD Operations
    def insert(self, root, patient):
        pass

    def delete(self, root, pid):
        pass

    def search(self, root, pid):
        pass

    # Traversal / Display Functions
    def inorder(self, root):
        pass

    def display_all_patients(self):
        pass

    def display_patients_with_disease(self, root, disease):
        pass

    def count_patients_doctor(self, root, doctor):
        pass

    def is_balanced(self, root):
        pass
```

Test Function

```
def test():
    hospital = HospitalAVL()

    # Insert sample patients
    p1 = Patient(101, "Alice", 30, "Female", "Flu", "Dr. A", "20-09-2025")
    p2 = Patient(102, "Bob", 45, "Male", "Malaria", "Dr. B", "18-09-2025")
    p3 = Patient(103, "Charlie", 60, "Male", "Diabetes", "Dr. C", "15-09-2025")
    p4 = Patient(104, "Daisy", 25, "Female", "Asthma", "Dr. A", "19-09-2025")
    p5 = Patient(105, "Ethan", 70, "Male", "Cancer", "Dr. D", "10-09-2025")

    hospital.root = hospital.insert(hospital.root, p1)
    hospital.root = hospital.insert(hospital.root, p2)
    hospital.root = hospital.insert(hospital.root, p3)
    hospital.root = hospital.insert(hospital.root, p4)
    hospital.root = hospital.insert(hospital.root, p5)

    # Test search
    print("Searching for PID 103:")
    hospital.search(hospital.root, 103)

    # Test display all patients
    print("\nAll patients (In-order):")
    hospital.display_all_patients()

    # Test display patients with a specific disease
    print("\nPatients with Malaria:")
    hospital.display_patients_with_disease(hospital.root, "Malaria")

    # Test counting patients under a specific doctor
    print("\nNumber of patients under Dr. A:")
    hospital.count_patients_doctor(hospital.root, "Dr. A")

    # Test delete
    print("\nDeleting PID 101 (Alice)...")
    hospital.root = hospital.delete(hospital.root, 101)

    # Display after deletion
    print("\nAll patients after deletion:")
    hospital.display_all_patients()

    # Check if AVL Tree is balanced
    print("\nCheck if AVL Tree is balanced:")
    if hospital.is_balanced(hospital.root):
        print("Tree is balanced")
    else:
        print("Tree is NOT balanced")

if __name__ == "__main__":
    test()
```

Sample Output

```
Searching for PID 103:
Found Patient      PID: 103, Name: Charlie, Age: 60, Gender: Male,
Disease: Diabetes, Doctor: Dr. C, Admission: 15-09-2025

All patients (In-order):
PID: 101, Name: Alice, Age: 30, Disease: Flu, Doctor: Dr. A
PID: 102, Name: Bob, Age: 45, Disease: Malaria, Doctor: Dr. B
PID: 103, Name: Charlie, Age: 60, Disease: Diabetes, Doctor: Dr. C
PID: 104, Name: Daisy, Age: 25, Disease: Asthma, Doctor: Dr. A
PID: 105, Name: Ethan, Age: 70, Disease: Cancer, Doctor: Dr. D

Patients with Malaria:
PID: 102, Name: Bob, Age: 45, Disease: Malaria, Doctor: Dr. B

Number of patients under Dr. A:
2

Deleting PID 101 (Alice)...

All patients after deletion:
PID: 102, Name: Bob, Age: 45, Disease: Malaria, Doctor: Dr. B
PID: 103, Name: Charlie, Age: 60, Disease: Diabetes, Doctor: Dr. C
PID: 104, Name: Daisy, Age: 25, Disease: Asthma, Doctor: Dr. A
PID: 105, Name: Ethan, Age: 70, Disease: Cancer, Doctor: Dr. D

Check if AVL Tree is balanced:
    Tree is balanced
```