Binary Search Tree:

class LogNode:

    def \_\_init\_\_(self, name, time, purpose):

        self.name = name

        self.time = time

        self.purpose = purpose

        self.left = None

        self.right = None

class LogBookBST:

    def \_\_init\_\_(self):

        self.root = None

    def insert(self, root, node):

        if root is None:

            return node

        if node.name < root.name:

            root.left = self.insert(root.left, node)

        else:

            root.right = self.insert(root.right, node)

        return root

    def add\_entry(self, name, time, purpose):

        new\_node = LogNode(name, time, purpose)

        if self.root is None:

            self.root = new\_node

        else:

            self.insert(self.root, new\_node)

        print(f"Entry for '{name}' added successfully.")

    def search(self, root, name):

        if root is None or root.name == name:

            return root

        if name < root.name:

            return self.search(root.left, name)

        else:

            return self.search(root.right, name)

    def find\_min(self, node):

        current = node

        while current.left is not None:

            current = current.left

        return current

    def delete(self, root, name):

        if root is None:

            print(f"No entry found for '{name}'.")

            return root

        if name < root.name:

            root.left = self.delete(root.left, name)

        elif name > root.name:

            root.right = self.delete(root.right, name)

        else:

            if root.left is None and root.right is None:

                return None

            elif root.left is None:

                return root.right

            elif root.right is None:

                return root.left

            temp = self.find\_min(root.right)

            root.name, root.time, root.purpose = temp.name, temp.time, temp.purpose

            root.right = self.delete(root.right, temp.name)

            print(f"Entry for '{name}' deleted successfully.")

        return root

    def remove\_entry(self, name):

        self.root = self.delete(self.root, name)

    def inorder(self, root):

        if root:

            self.inorder(root.left)

            print(f"{root.name} - {root.time} - {root.purpose}")

            self.inorder(root.right)

    def preorder(self, root):

        if root:

            print(f"{root.name} - {root.time} - {root.purpose}")

            self.preorder(root.left)

            self.preorder(root.right)

    def postorder(self, root):

        if root:

            self.postorder(root.left)

            self.postorder(root.right)

            print(f"{root.name} - {root.time} - {root.purpose}")

    def count\_entries(self, root):

        if root is None:

            return 0

        return 1 + self.count\_entries(root.left) + self.count\_entries(root.right)

if \_\_name\_\_ == "\_\_main\_\_":

    logbook = LogBookBST()

    while True:

        print("\n--- LogBook Menu ---")

        print("1. Add a new entry")

        print("2. Search for an entry")

        print("3. Remove an entry")

        print("4. Display all entries (Inorder)")

        print("5. Count total entries")

        print("6. Exit")

        choice = input("Enter your choice: ")

        match choice:

            case '1':

                name = input("Enter visitor name: ")

                time = input("Enter time of visit: ")

                purpose = input("Enter purpose of visit: ")

                logbook.add\_entry(name, time, purpose)

            case '2':

                name = input("Enter name to search: ")

                s = logbook.search(logbook.root, name)

                if s:

                    print(f"Found: Name: {s.name}, Time: {s.time}, Purpose: {s.purpose}")

                else:

                    print(f"Entry for '{name}' not found.")

            case '3':

                name = input("Enter name to remove: ")

                logbook.remove\_entry(name)

            case '4':

                print("\n--- All LogBook Entries (Sorted by Name) ---")

                logbook.inorder(logbook.root)

            case '5':

                total = logbook.count\_entries(logbook.root)

                print(f"Total entries in the LogBook: {total}")

            case '6':

                print("Exiting LogBook application. Goodbye!")

                break

            case \_:

                print("Invalid choice. Please try again.")

Output:

--- LogBook Menu ---

1. Add a new entry

2. Search for an entry

3. Remove an entry

4. Display all entries (Inorder)

5. Count total entries

6. Exit

Enter your choice: 1

Enter visitor name: Jeebi

Enter time of visit: 2

Enter purpose of visit: function

Entry for 'Jeebi' added successfully.

--- LogBook Menu ---

1. Add a new entry

2. Search for an entry

3. Remove an entry

4. Display all entries (Inorder)

5. Count total entries

6. Exit

Enter your choice: 1

Enter visitor name: XXX

Enter time of visit: 3

Enter purpose of visit: function

Entry for 'XXX' added successfully.

--- LogBook Menu ---

1. Add a new entry

2. Search for an entry

3. Remove an entry

4. Display all entries (Inorder)

5. Count total entries

6. Exit

Enter your choice: 1

Enter visitor name: GB

Enter time of visit: 4

Enter purpose of visit: function

Entry for 'GB' added successfully.

--- LogBook Menu ---

1. Add a new entry

2. Search for an entry

3. Remove an entry

4. Display all entries (Inorder)

5. Count total entries

6. Exit

Enter your choice: 2

Enter name to search: GB

Found: Name: GB, Time: 4, Purpose: function

--- LogBook Menu ---

1. Add a new entry

2. Search for an entry

3. Remove an entry

4. Display all entries (Inorder)

5. Count total entries

6. Exit

Enter your choice: 3

Enter name to remove: GB

--- LogBook Menu ---

1. Add a new entry

2. Search for an entry

3. Remove an entry

4. Display all entries (Inorder)

5. Count total entries

6. Exit

Enter your choice: 4

--- All LogBook Entries (Sorted by Name) ---

Jeebi - 2 - function

XXX - 3 - function

--- LogBook Menu ---

1. Add a new entry

2. Search for an entry

3. Remove an entry

4. Display all entries (Inorder)