Phone Directory System Project Documentation

Batch No.: 27

Reg No.s: 205002118, 205002119

Names: Vaishnavi Anand, Vandhana Vishnuraj

**Problem Description:**

This project implements a phone's contact directory (a listing of telephone numbers) using doubly linked lists, splay and AVL trees. The project will demonstrate searching for a particular telephone number, finding and deleting duplicates (when inserting into a splay tree), adding new telephone numbers and updating any given number. The main aim of this project is to create an optimized phone directory system. Additionally, it shows the most recently used contact(s) through splay tree traversal and listing of contacts in alphabetic and reverse order.

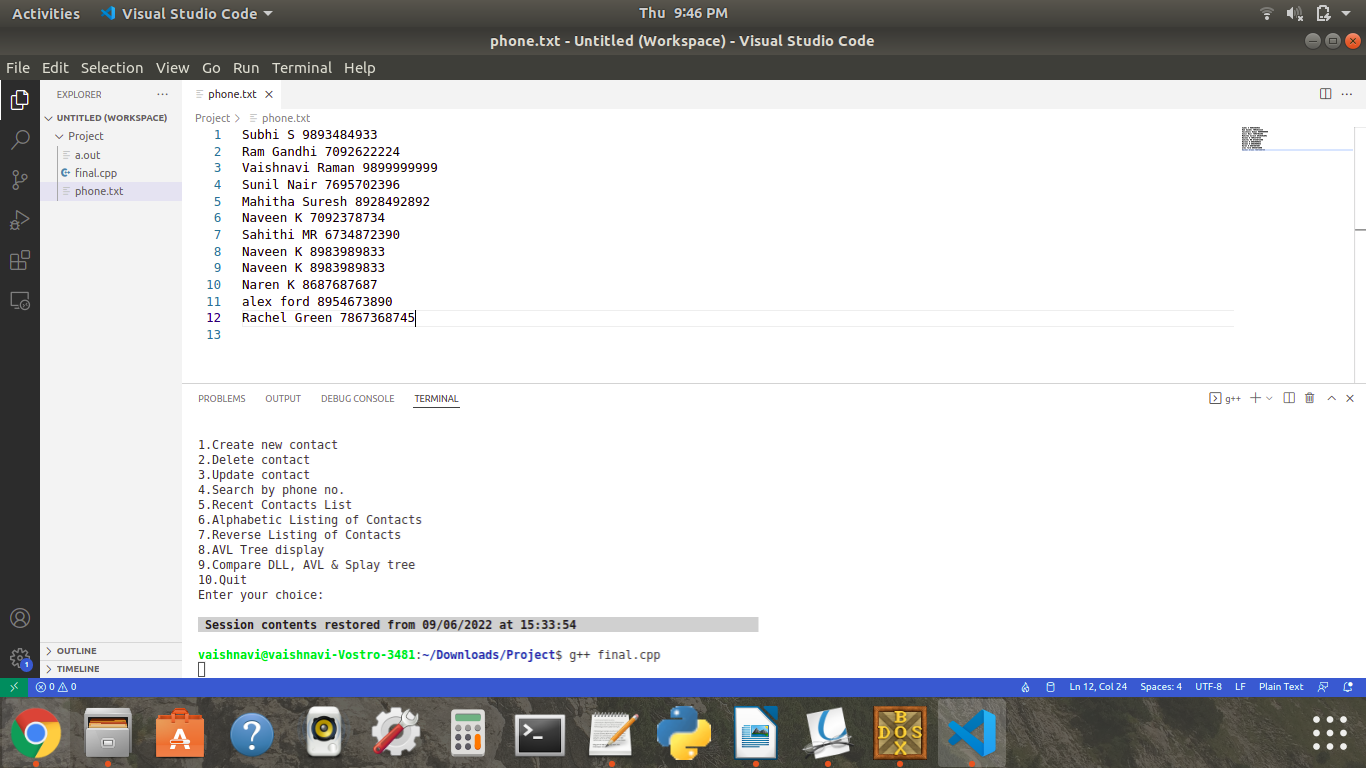
This application focuses on CRUD operations for a phone's contact directory. Faster searching and avoiding redundancy are the most beneficial features for a user.

Data stored in the doubly linked list nodes will be a contact's first, last names and phone number (of 10 digits). Data is read from the existing csv file (text separated) and stored in a splay tree. Doubly linked lists and AVL trees are constructed simultaneously, showing all changes after each operation. New input is appended to the file.

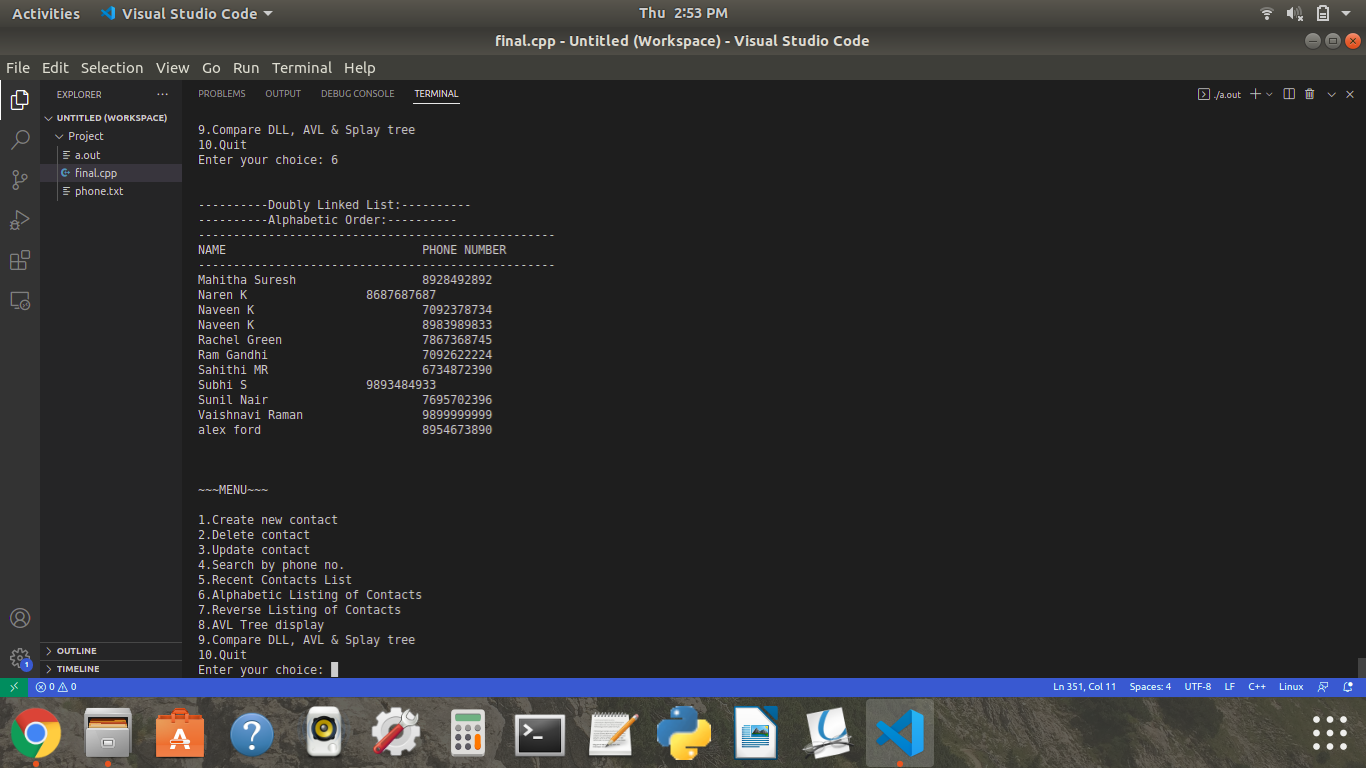
Since a phone contact list is in sorted order, and we need optimized searching implementation, the existing DLL is sorted using a merge sort algorithm.

To compare the insertions in DLL, AVL tree and splay tree, we use time.h header file to find the time taken in each function and display comparison.

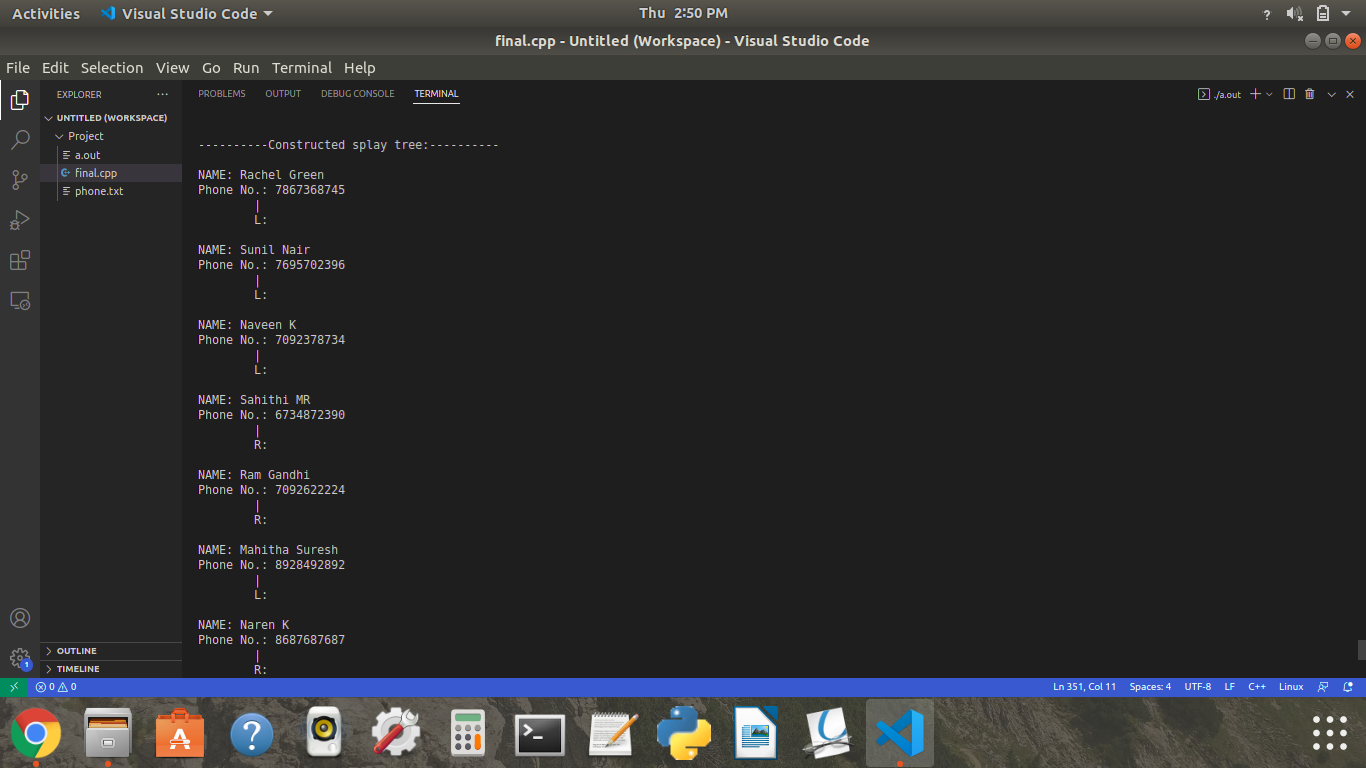
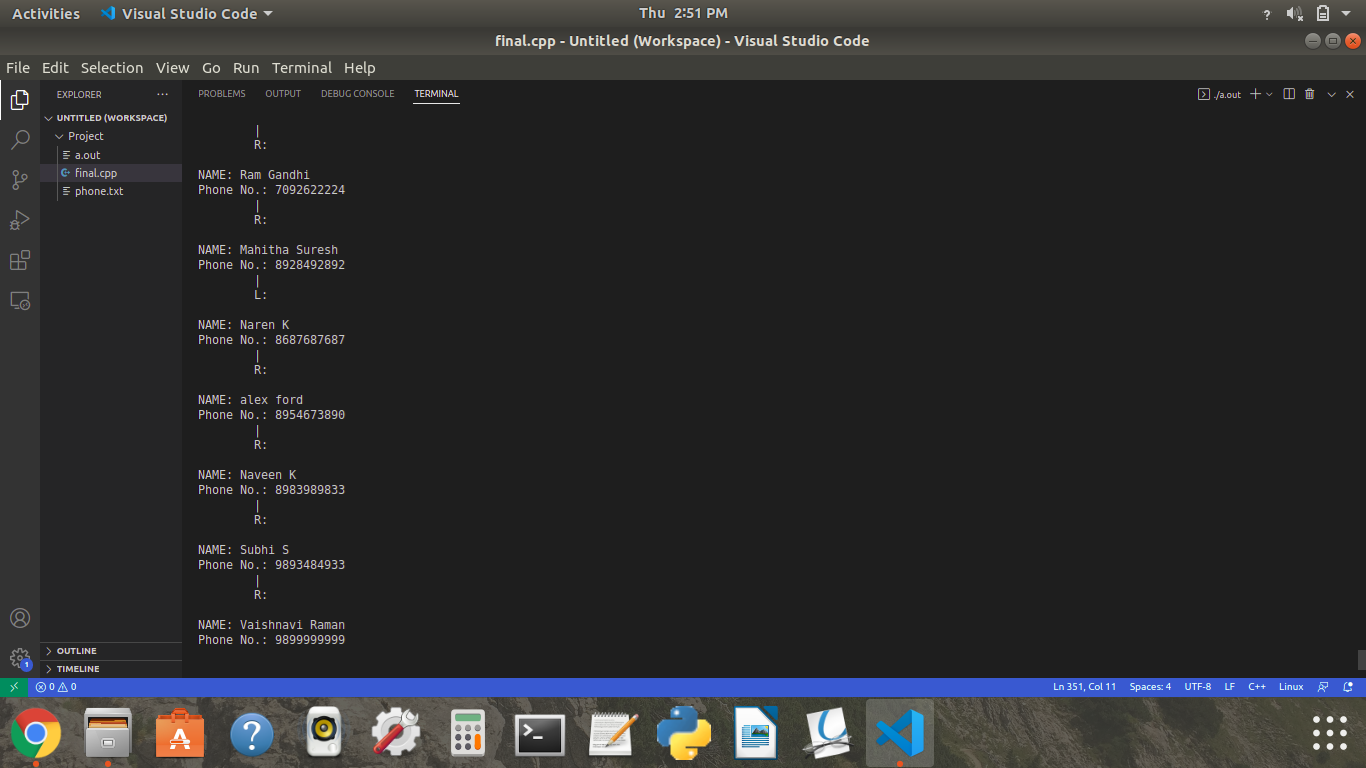
Data file:

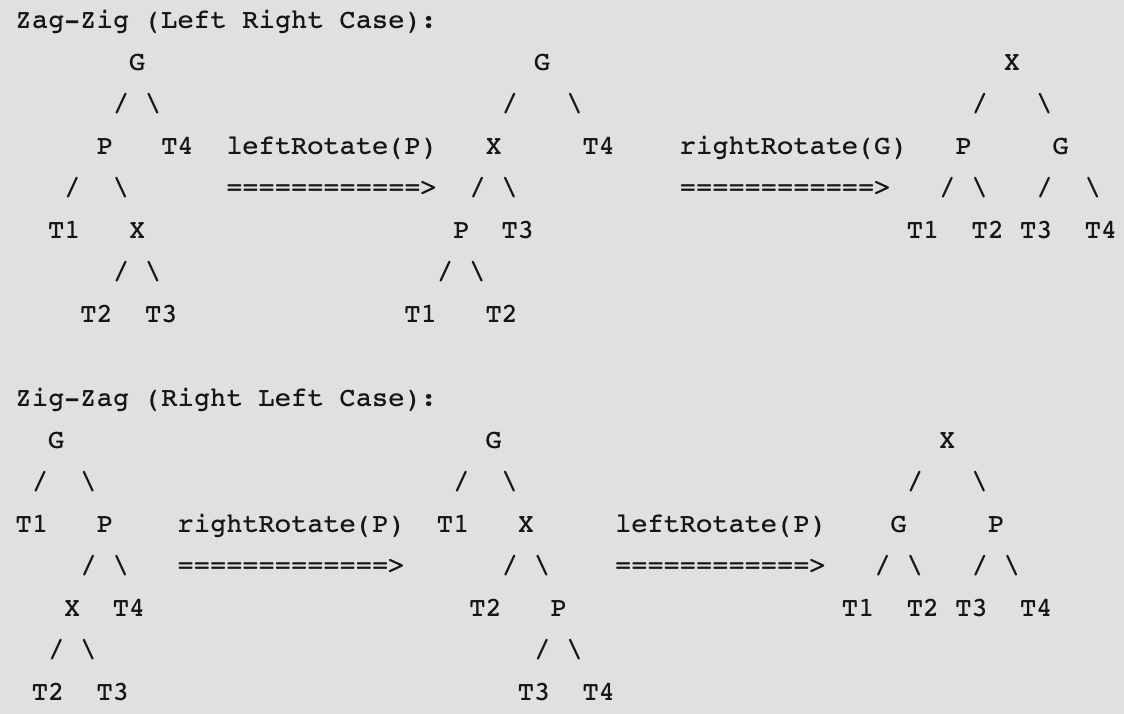


DLL:

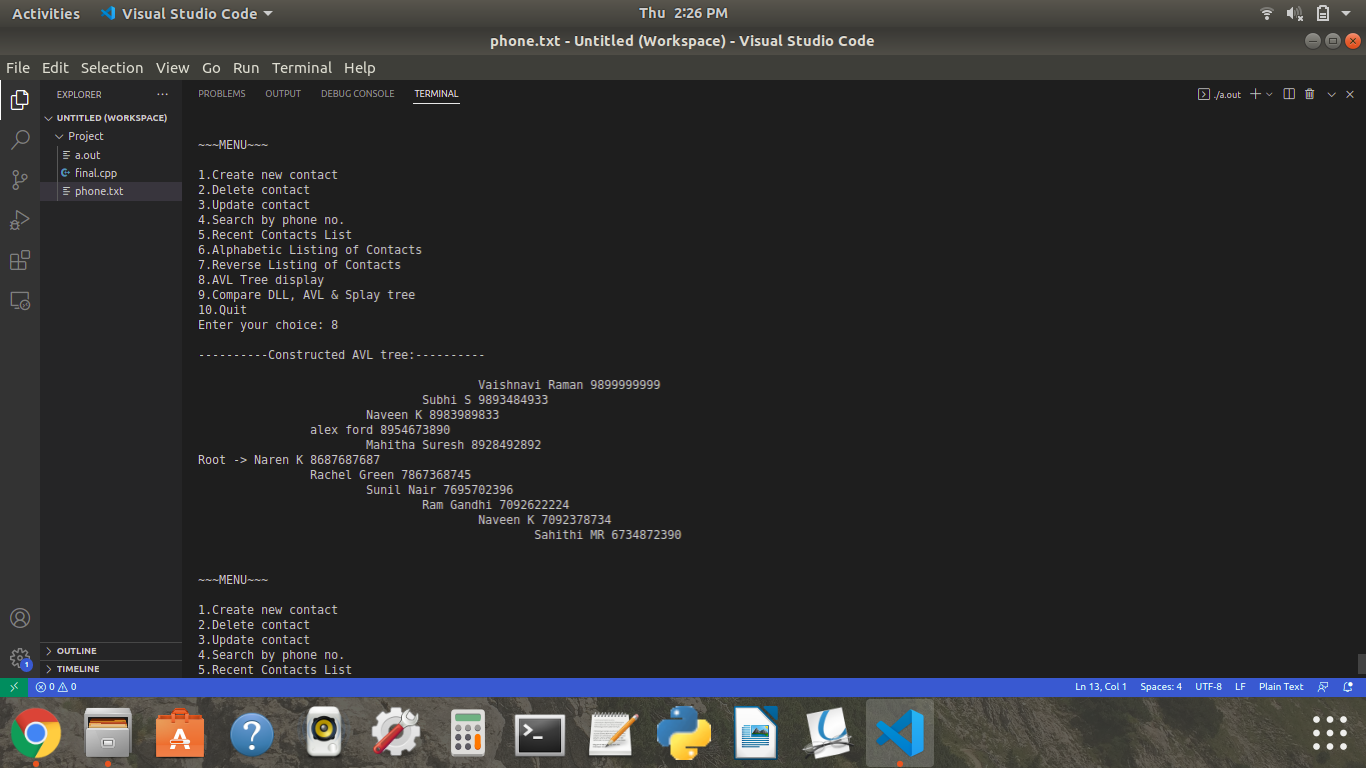


Splay Tree:



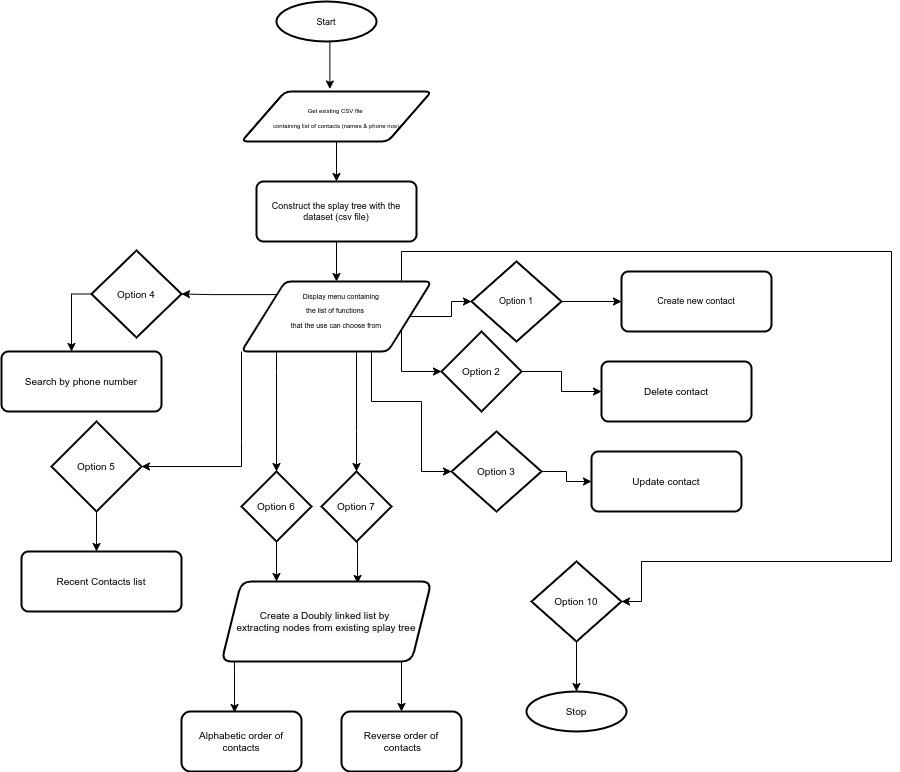
AVL Tree:



Within the search tree, each node contains a person's basic information.

The tree can be constructed based on the phone number.

**Flow Diagram:**



**Result:**

Thus a program that implements a phone directory system with CRUD operations and features like alphabetical listing, recently accessed contact list and time calculation to compare insertion and search is successfully completed.

**Inference:**

On developing our project, we tested our project with three different types of data structures:

1. Splay trees
2. Doubly linked list
3. AVL trees

On comparing the average time taken for inserting all contacts from a data file (nodes) into the data structure (Splay trees, DLL, AVL Trees), we infer that insertion in a splay tree takes the least time. However, if the number of records were larger, DLL insertion would be more efficient.

On comparing the search time with AVL and splay tree, we infer that searching for a phone number in an AVL tree takes more time than splay tree.

Time taken comparison in 1 execution of application:

