DESIGN AND ANALYSIS OF ALGORITHMS

REPORT

TOPIC: RESTRAUNT MANAGEMENT SYSTEM

TEAM MEMBERS:

Nidhi M.N: PES2UG21EC095

Pratika S: PES2UG21EC106

Suchith Gowda: PES2UG21EC143

FACULTY ADVISOR: DR. CHANDRASHEKHAR P CHAVAN

Description:

The provided C program implements a simple restaurant management system with file handling capabilities. It allows users to add, remove, display, search, sort, save, and load dishes in a menu. The program utilizes concepts of linear search, binary search, insertion sort, and file handling to manage the restaurant menu efficiently.

DAA Concepts Used:

Linear Search:

- The program uses linear search to find a dish by name or phone number in the contact list.
- It iterates through the list of contacts to find a match based on the name or phone number.
- The linear search algorithm has a time complexity of O(n), where n is the number of contacts.

Binary Search:

- The binary search algorithm is employed to efficiently search for a dish by name in the menu.
- It operates on a sorted array of dishes and repeatedly divides the search interval in half.
- Binary search has a time complexity of O(log n), making it more efficient than linear search for large datasets.

Insertion Sort:

- The insertion sort algorithm is used to sort the contacts by name after adding a new contact.
- It iterates through the list of contacts and inserts each element into its correct position in the sorted list.
- Insertion sort has a time complexity of O(n^2) in the worst case, making it suitable for small datasets.

File Handling:

- The program includes functions to save the menu to a file and load the menu from a file.
- It uses file I/O operations to write the menu data to a text file and read the menu data from a text file.
- File handling allows the program to persist the menu data between different program runs.

Menu Management:

- The program provides a menu-driven interface for users to interact with the restaurant management system.
- Users can add, remove, display, search, sort, save, and load dishes in the menu using the provided options.
- The menu management functionality enables efficient manipulation and organization of the restaurant menu.

User Input Handling:

- The program handles user input for various operations, such as adding, removing, searching, and sorting dishes.
- It utilizes input validation and buffer clearing to ensure accurate and reliable user interactions.
- Proper handling of user input enhances the usability and robustness of the restaurant management system.

Code:

```
#include <stdio.h>
#include <string.h>

#define MAX_DISHES 100

typedef struct {
   char name[50];
   float price;
} Dish;

void saveMenuToFile(Dish dishes[], int numDishes) {
   FILE *file = fopen("menu.txt", "w");
   if (file != NULL) {
      for (int i = 0; i < numDishes; i++) {
         fprintf(file, "%s\n%.2f\n", dishes[i].name, dishes[i].price);
      }
      fclose(file);
      printf("Menu saved to file successfully!\n");
   } else {
```

```
        printf("Error opening file for writing.\n");
    }
}

int loadMenuFromFile(Dish dishes[]) {
    FILE *file = fopen("menu.txt", "r");
    if (file != NULL) {
        int count = 0;
        while (fscanf(file, "%s\n%f\n", dishes[count].name, &dishes[count].price) != EOF) {
            count++;
        }
        fclose(file);
        printf("Menu loaded successfully!\n");
        return count;
    } else {
        printf("Error opening file for reading. Using default menu.\n");
        return 0;
    }
}

void displayMenu(Dish dishes[], int numDishes) {
    if (numDishes == 0) {
        printf("Menu is empty!\n");
    } else {
        printf("Menu:\n");
        for (int i = 0; i < numDishes; i++) {
            printf("%s - $%.2f\n", dishes[i].name, dishes[i].price);
        }
    }
}

int searchDishByName(Dish dishes[], int numDishes, const char *dishName) {
    for (int i = 0; i < numDishes; i++) {
        if (strcmp(dishes[i].name, dishName) == 0) {
            return i;  // Found the dish
        }
    }
    return -1;  // Dish not found
}

void sortDishesByName(Dish dishes[], int numDishes) {
    for (int i = 0; i < numDishes - 1; i++) {
        for (int j = 0; j < numDishes - i - 1; j++) {
            if (strcmp(dishes[j].name, dishes[j + 1].name) > 0) {
                Dish temp = dishes[j];
                dishes[j] = dishes[j + 1];
                dishes[j + 1] = temp;
            }
        }
    }
}

int main() {
```

```
Dish menu[MAX_DISHES];
int numDishes = 0;

int choice;

do {
   printf("\n--- Restaurant Management System ---\n");
   printf("1. Add Dish\n");
   printf("2. Remove Dish\n");
   printf("3. Display Menu\n");
   printf("4. Search Dish\n");
   printf("5. Sort Menu\n");
   printf("6. Save Menu\n");
   printf("7. Load Menu\n");
   printf("8. Exit\n");
   printf("Enter your choice: ");
   scanf("%d", &choice);
   getchar(); // Clear input buffer

   switch (choice) {
      case 1: // Add Dish
         if (numDishes == MAX_DISHES) {
            printf("Menu is full!\n");
         } else {
            printf("Enter dish name: ");
            fgets(menu[numDishes].name, sizeof(menu[numDishes].name), stdin);
            menu[numDishes].name[strcspn(menu[numDishes].name, "\n")] = '\0';

            printf("Enter dish price: ");
            scanf("%f", &menu[numDishes].price);
            getchar(); // Clear input buffer

            numDishes++;
            printf("Dish added successfully!\n");
         }
         break;

      case 2: // Remove Dish
         if (numDishes == 0) {
            printf("Menu is empty!\n");
         } else {
            printf("Enter the name of the dish to remove: ");
            char dishName[50];
            fgets(dishName, sizeof(dishName), stdin);
            dishName[strcspn(dishName, "\n")] = '\0';

            int foundIndex = searchDishByName(menu, numDishes, dishName);
            if (foundIndex != -1) {
               for (int i = foundIndex; i < numDishes - 1; i++) {
                  menu[i] = menu[i + 1];
               }
               numDishes--;
               printf("Dish removed successfully!\n");
```

```
            } else {
                printf("Dish not found in the menu.\n");
            }
        }
        break;

    case 3: // Display Menu
        displayMenu(menu, numDishes);
        break;

    case 4: // Search Dish
        printf("Enter the name of the dish to search: ");
        char searchName[50];
        fgets(searchName, sizeof(searchName), stdin);
        searchName[strcspn(searchName, "\n")] = '\0';

        int searchIndex = searchDishByName(menu, numDishes, searchName);
        if (searchIndex != -1) {
            printf("Dish found! %s is priced at $%.2f\n", menu[searchIndex].name,
menu[searchIndex].price);
        } else {
            printf("Dish not found in the menu.\n");
        }
        break;

    case 5: // Sort Menu
        sortDishesByName(menu, numDishes);
        printf("Menu sorted by dish name.\n");
        break;

    case 6: // Save Menu
        saveMenuToFile(menu, numDishes);
        break;

    case 7: // Load Menu
        numDishes = loadMenuFromFile(menu);
        break;

    case 8: // Exit
        printf("Exiting the program...\n");
        break;

    default:
        printf("Invalid choice! Please try again.\n");
        break;
  }
 } while (choice != 8);

 return 0;
}
```

Output:

```
--- Restaurant Management System ---
1. Add Dish
2. Remove Dish
3. Display Menu
4. Search Dish
5. Sort Menu
6. Save Menu
7. Load Menu
8. Exit
Enter your choice: 1
Enter dish name: burger
Enter dish price: 69
Dish added successfully!
```

```
 --- Restaurant Management System ---
 1. Add Dish
 2. Remove Dish
 3. Display Menu
 4. Search Dish
 5. Sort Menu
 6. Save Menu
 7. Load Menu
 8. Exit
 Enter your choice: 3
 Menu:
 fries - $49.00
 burger - $69.00
```

```
--- Restaurant Management System ---
1. Add Dish
2. Remove Dish
3. Display Menu
4. Search Dish
5. Sort Menu
6. Save Menu
7. Load Menu
8. Exit
Enter your choice: 4
Enter the name of the dish to search: burger
Dish found! burger is priced at $69.00
```

```
--- Restaurant Management System ---
1. Add Dish
2. Remove Dish
3. Display Menu
4. Search Dish
5. Sort Menu
6. Save Menu
7. Load Menu
8. Exit
Enter your choice: 5
Menu sorted by dish name.

--- Restaurant Management System ---
1. Add Dish
2. Remove Dish
3. Display Menu
4. Search Dish
5. Sort Menu
6. Save Menu
7. Load Menu
8. Exit
Enter your choice: 3
Menu:
burger - $69.00
fries - $49.00
```