RESTAURANT MANAGEMENT SYSTEM

A PROJECT REPORT

SUBMITTED IN COMPLETE FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by:

DEVVART

2K20/IT/41

DHARAMVEER

2K20/IT/42

Under the supervision

Of

DR. RITU AGARWAL

DELHI TECHNOLOGICAL UNIVERSITY

Bawana Road, Delhi-110042

DECLARATION

We hereby certify that the work, which is presented in the project, entitled "***RESTAURANT MANAGEMENT SYSTEM***". Is in fulfilment of the innovative project required for the evaluation of the first odd semester and submitted to the Department of Information Technology, Delhi Technological University, under the supervision of Dr. Ritu Agarwal.

The work presented in this report has not been submitted and not under consideration for the award for any other course/degree of this or any other Institute/University.

DEVVART
2k20/IT/41
B.Tech, Information Technology

DHARAMVEER
2k20/IT/42
B.Tech, Information Technology

CERTIFICATE

I hereby certify that the project Dissertation titled "***RESTAURANT MANAGEMENT SYSTEM***" which is submitted by DEVVART (2K20/IT/41) DHARAMVEER (2K20/IT/42) [INFORMATION TECHNOLOGY], Delhi Technological University, Delhi in complete fulfilment of the requirement for the award of the degree of the Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi **Dr. RITU AGARWAL**

Date: 11/11/2021 **(Teacher In Charge)**

ACKNOWLEDGEMENT

In performing our major project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much pleasure. We would like to show our gratitude **Dr. Ritu Agarwal,** Mentor for major project. Giving us a good guideline for report throughout numerous consultations. We would also like to extend our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment.

My classmates have made valuable comment suggestions on this proposal which gave me an inspiration to improve my assignment. We thank all the people for their help directly and indirectly to complete our assignment.

In addition, we would like to thank Department of Information Technology, Delhi Technological University for giving us the opportunity to work on this topic.

INTRODUCTION

From the origin of the first programming languages to the modern programming languages currently in use, computer programming has evolved quite a lot. It has now become more powerful, efficient, and advanced. However, the fundamental concepts and use of data structure and algorithms in computer programming have not changed. DSA has been the core of computer programming from the beginning.

You might have heard DSA being used mainly in the field of computer science. However, the use of DSA is not limited to the field of computing. We can also find the concept of DSA being used in day-to-day life. In this project, we will discuss the common concept of DSA that is used in everyday life. But before that, let's learn the basics of Data Structure and Algorithms first.

Data structure and algorithms is a branch of computer science that deals with creating machine-efficient and optimized computer programs. The term **Data Structure** refers to the storage and organization of data, and Algorithm refers to the step-by-step procedure to solve a problem.

In this project we have tried to bring about the importance and the influence of these important Data Structures like Arrays, Strings, Stacks, Queues, Linked Lists, Graphs and many algorithms like sorting, searching, insertion and deletion in a very common real life scenario of event management, We have considered here the case of a peaceful drive with the motive to open offline college for more quality education and student development.

We have managed to depict the advantage and the application of various above mentioned Data structures in the proper data management during an event.
Our project works of a user-friendly menu driven interface which allows the user to perform various helpful and necessary operation within fraction of seconds and with ease. The versatility of our code makes it so special it can be made to use to manage competitions, classroom record, office record, maintenance work, volunteer work, hospital management, reservation systems and many more.

Diving deeper into the main essence of the project, we have made a live scenario of a college rally where various tasks such as getting data of volunteers, searching volunteers, removing volunteers, sorting volunteers, managing the funds department queues, managing advertising department, rally routes and many more. These tasks have been brought to life by the help of Data Structures and why it is necessary is the core motive of this project.

We worked on the applicability and user-friendly environment to an extent where the interface is not only simple to understand but highly efficient and structured.
This project is original and was inspired by online college events conducted by us.

INTRODUCTION TO PROJECT

A restaurant management system is a type of point-of-sale (POS) software specifically designed for restaurants, bars, food trucks and others in the food service industry.

BENEFITS OF RMS

- Track sales and orders
- View accurate, real-time financial statements
- Access data easily and faster

FEATURES

EASY HANDLING

Restaurant management system is designed to handle all the primary information required to calculate such as final bills, total sales during the entire day.

INTERACTIVE

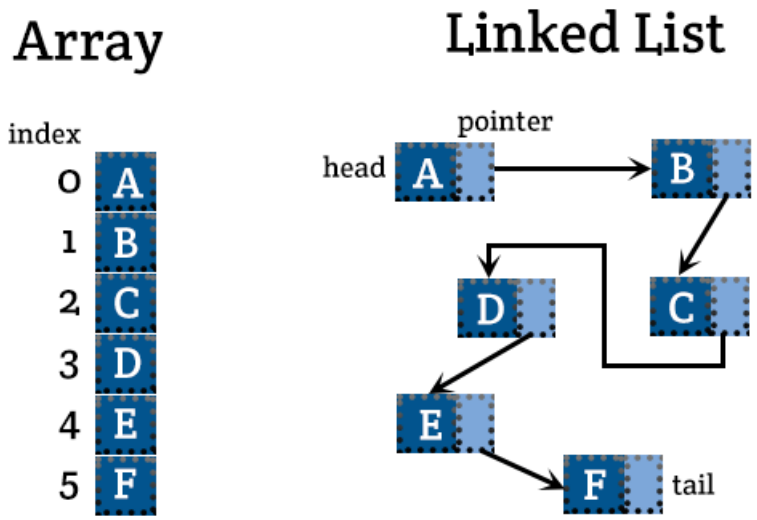
The main purpose of the Restaurant Management System is to reach to wider range of customers and to educate them about existing and new items offered by restaurants.

LINKED LIST AS DATABASE

As we will using linked list as database in this restaurant management system, we will have all linked list advantages such as we can grow or shrink it any time as per our menu as it is a dynamic data structure

ARRAY VS LINKED LIST

Arrays store elements in contiguous memory locations, resulting in easily calculable addresses for the elements stored and this allows faster access to an element at a specific index. Linked lists are less rigid in their storage structure and elements are usually not stored in contiguous locations; hence they need to be stored with additional tags giving a reference to the next element. This difference in the data storage scheme decides which data structure would be more suitable for a given situation

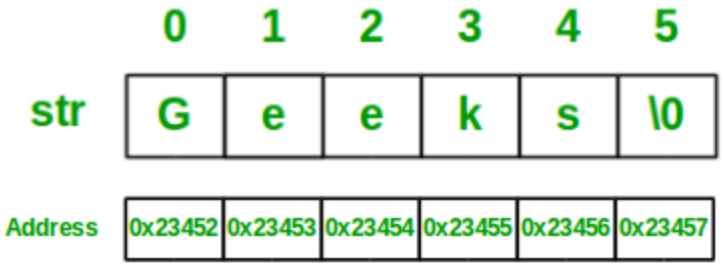


STRUCTURE AND COMPONENTS OF RMS

Before Getting into the code, we will be discussing the Data structures used and their theory:

Dynamic memory allocation: The concept of dynamic memory allocation in c language enables the C programmer to allocate memory at runtime. Dynamic memory allocation in c language is possible by 4 functions of stdlib.h header file.

String: a string is a sequence of characters terminated with a null character \0 . For example: char c[] = "c string"; When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end by default.



LINKED LIST: A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. ... In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

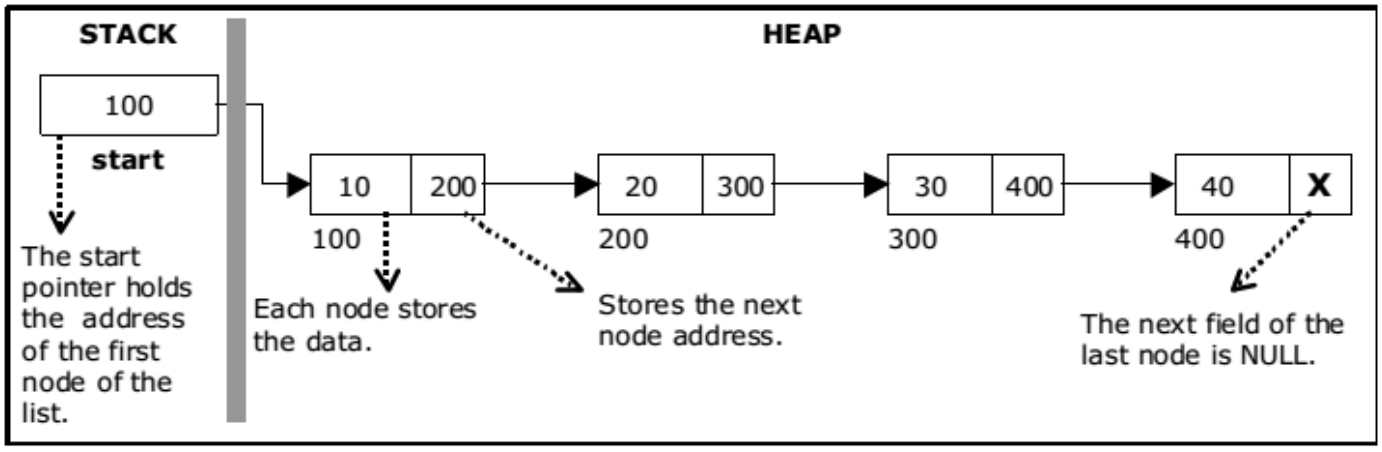
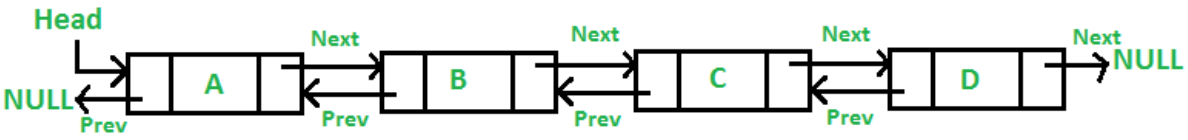


Figure 3.2.1. Single Linked List

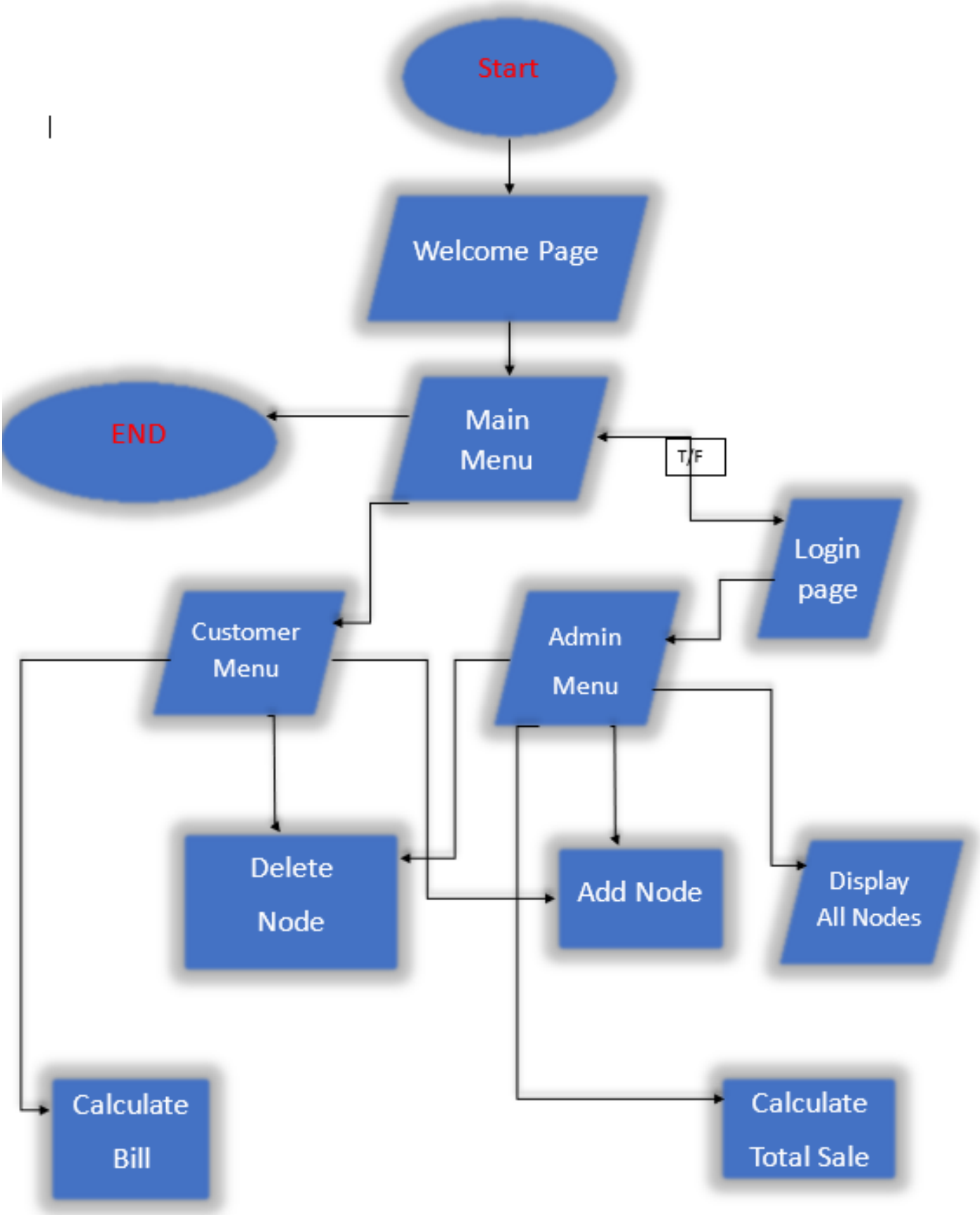
DOUBLY LINKED LISTS: - Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence. Therefore, in a doubly linked list, a node consists of three parts: node data, pointer to the next node in sequence (next pointer) , pointer to the previous node (previous pointer)..



Header files Included

- Conio.h
- Stdio.h
- String.h
- Window.h
- Time.h

Flowchart


Start
Welcome Page
Main Menu
END
T/F
Login page
Customer Menu
Admin Menu
Delete Node
Add Node
Display All Nodes
Calculate Bill
Calculate Total Sale

Code with Explanation

```
/*
This code is written by:

1. 2k20/IT/41 - Devvart
2. 2k20/IT/42 - Dharamveer

*/

#include<stdio.h> //This header file includes functions involving memory allocation,
process control, conversions and others.
#include<string.h>    //This header file includes string functions.
#include <windows.h> // This header file includes Sleep() function to pause screen.
#include <time.h>   // To use time() function
#include <conio.h>   //This header file provides basic I/O operations for program

//In this program we have implemented Linked List to perform various tasks.
struct node //This is the structure which provides a template for node of Linked List. It
shows a representation of a food item in men
{                                                  // It contains different attributes of node
such as it's food name, gauntity, price, data, pointer to next node and pointer to
previous node.
    char foodname[50];
    int quantity;
    float price;
    int data;
    struct node *prev;
    struct node *next;
};
//Here in order to perform various task of admin as well as customer using one single
structure, we have created one single struct

//catering to purpose of menu item as well as customer's order.


struct node *headc = NULL, *newnode, *tailc = NULL;      //global struct pointers which
are used throughout the program to create linked list and maintain it.
struct node *heada = NULL, *taila = NULL;
struct node *head_s;

void adminmenu()
{
    printf("\n\t\t\t\t\t\t\t1. View total sales\n");
    printf("\t\t\t\t\t\t\t2. Add new items in the order menu\n");
    printf("\t\t\t\t\t\t\t3. Delete items from the order menu\n");
    printf("\t\t\t\t\t\t\t4. Display order menu\n");
    printf("\t\t\t\t\t\t\t5. Back To Main Menu \n\n");
    printf("\t\t\t\t\t\t\t   Enter Your Choice --->");
}

void customermenu()
```

```
{
    printf("\n\t\t\t\t\t\t\t1. Place your order\n");
    printf("\t\t\t\t\t\t\t2. View your ordered items\n");
    printf("\t\t\t\t\t\t\t3. Delete an item from order\n");
    printf("\t\t\t\t\t\t\t4. Display final bill\n");
    printf("\t\t\t\t\t\t\t5. Back To Main Menu \n\n");
    printf("\t\t\t\t\t\t\t   Enter Your Choice --->");
}

struct node *createadmin(struct node *head, int data, char foodname[25], float price)
{
    newnode = (struct node *)malloc(sizeof(struct node));

    newnode->data = data;
    newnode->price = price;
    newnode->quantity = 0;
    strcpy(newnode->foodname, foodname);
    newnode->next = NULL;
    newnode->prev = NULL;

    struct node *temp = head;

    if (temp == NULL)
        heada = taila = newnode;
    else
    {
        while (temp->next != NULL)
            temp = temp->next;

        temp->next = newnode;
        newnode->prev = taila;
        taila = newnode;
    }

    return heada;
}

struct node *createcustomer(struct node *head, int data, int quantity)
{
    newnode = (struct node *)malloc(sizeof(struct node));

    struct node *temp1 = heada;
    int flag = 0;
    while (temp1 != NULL)
    {
        if (temp1->data == data)
        {
            flag = 1;
            break;
        }
        temp1 = temp1->next;
    }

    if (flag == 1)
```

```
    {
        newnode->data = data;
        newnode->price = quantity * (temp1->price);
        newnode->quantity = quantity;
        strcpy(newnode->foodname, temp1->foodname);
        newnode->next = NULL;
        newnode->prev = NULL;

        struct node *temp = head;

        if (temp == NULL)
            headc = tailc = newnode;
        else
        {
            while (temp->next != NULL)
                temp = temp->next;

            temp->next = newnode;
            newnode->prev = tailc;
            tailc = newnode;
        }
    }
    else
    {
        printf("\n\t\t\t\t\t\t\tThis item is not present in the menu!\n");
    }
    return headc;
}

void displayList(struct node *head)
{
    struct node *temp1 = head;
    if (temp1 == NULL)
    {
        printf("\n\t\t\t\t\t\t\t\tList is empty!!\n\n");
    }
    else
    {
        printf("\n");
        while (temp1 != NULL)
        {
            if (temp1->quantity == 0)
                printf("\t\t\t\t\t\t\t%d\t%s\t%0.2f\n", temp1->data, temp1->foodname,
temp1->price);
            else
            {
                printf("\t\t\t\t\t\t\t%d\t%s\t%d\t%0.2f\n", temp1->data, temp1->foodname,
temp1->quantity, temp1->price);
            }

            temp1 = temp1->next;
        }
        printf("\n");
    }
```

```
}

struct node *totalsales(int data, int quantity)
{
    newnode = (struct node *)malloc(sizeof(struct node));
    int flag = 0;

    struct node *temp1 = heada;
    while (temp1->data != data)
    {
        temp1 = temp1->next;
    }

    newnode->data = data;
    newnode->price = quantity * (temp1->price);
    newnode->quantity = quantity;
    strcpy(newnode->foodname, temp1->foodname);
    newnode->next = NULL;
    newnode->prev = NULL;

    struct node *temp = head_s;

    if (temp == NULL)
        head_s = newnode;
    else
    {
        while (temp->next != NULL)
        {
            if (temp->data == data)
            {
                flag = 1;
                break;
            }
            temp = temp->next;
        }

        if (flag == 1)
        {
            temp->quantity += newnode->quantity;
            temp->price += newnode->price;
        }
        else
        {
            temp->next = newnode;
        }
    }

    return head_s;
}

void calculatetotsales()
{
    struct node *temp = headc;
    while (temp != NULL)
```

```
    {
        head_s = totalsales(temp->data, temp->quantity);
        temp = temp->next;
    }
}

struct node *delete (int data, struct node *head, struct node *tail)
{
    if (head == NULL)
    {
        printf("\n\t\t\t\t\t\t\tList is empty\n");
    }
    else
    {
        struct node *temp;
        if (data == head->data)
        {
            temp = head;
            head = head->next;
            if (head != NULL)
                head->prev = NULL;
            free(temp);
        }
        else if (data == tail->data)
        {
            temp = tail;
            tail = tail->prev;
            tail->next = NULL;
            free(temp);
        }
        else
        {
            temp = head;
            while (data != temp->data)
            {
                temp = temp->next;
            }
            (temp->prev)->next = temp->next;
            (temp->next)->prev = temp->prev;
            free(temp);
        }
    }
    return head;
}

int deleteadmin()
{
    printf("\n\t\t\t\t\tEnter serial no. of the food item which is to be deleted: ");
    int num;
    scanf("%d", &num);

    struct node *temp = heada;
    while (temp != NULL)
    {
```

```
        if (temp->data == num)
        {
            heada = delete (num, heada, taila);
            return 1;
        }
        temp = temp->next;
    }

    return 0;
}

int deletecustomer()
{
    printf("\n\t\t\t\t\tEnter serial no. of the food item which is to be deleted: ");
    int num;
    scanf("%d", &num);

    struct node *temp = headc;
    while (temp != NULL)
    {
        if (temp->data == num)
        {
            headc = delete (num, headc, tailc);
            return 1;
        }
        temp = temp->next;
    }

    return 0;
}

void displaybill()
{
    displayList(headc);
    struct node *temp = headc;
    float total_price = 0;
    while (temp != NULL)
    {
        total_price += temp->price;
        temp = temp->next;
    }

    printf("\t\t\t\t\t\tTotal price: %0.02f\n", total_price);
}

struct node *deleteList(struct node *head)
{
    if (head == NULL)
    {
        return NULL;
    }
    else
    {
        struct node *temp = head;
```

```
        while (temp->next != 0)
        {
            temp = temp->next;
            free(temp->prev);
        }
        free(temp);
        head = NULL;
    }

    return head;
}
int password()
{
    char cname[30], pass[20];
    int ch, i = 0, cap = 0, capt = 0;
    printf("\n\n\n\n\n\t\t\t\t\tUSER NAME:  ");
    fflush(stdin);
    gets(cname);
    printf("\n\t\t\t\t\tPASSWORD:    ");
    while ((ch = getch()) != 13)
    {
        printf("*");
        pass[i] = ch;
        i++;
    }
    pass[i] = '\0';
    srand(time(0));
    cap = rand();
    printf("\n\t\t\t\t\t CAPTURE:->  %d \n", cap);
    printf(" Please enter the valid capture :-   ");
    scanf("%d", &capt);
    if ((strcmp(cname, "a") == 0) && (strcmp(pass, "b") == 0) && cap == capt)
        return 1;
    else
        return 0;
}
void admin()
{
    system("cls");

    printf("\n\t\t\t\t\t   ----------------------------------------------\n");
    printf("\t\t\t\t\t\t    ADMIN SECTION\n");
    printf("\t\t\t\t\t   ----------------------------------------------\n");
    while (1)
    {

        adminmenu();

        int opt;
        scanf("%d", &opt);

        if (opt == 5)
            break;
```

```
        switch (opt)
        {
        case 1:
        {
            system("cls");
            displayList(head_s);
            break;
        }
        case 2:
            system("cls");
            printf("\n\t\t\t\t\t\t\tEnter serial no. of the food item: ");
            int num, flag = 0;
            char name[50];
            float price;
            scanf("%d", &num);

            struct node *temp = heada;

            while (temp != NULL)
            {
                if (temp->data == num)
                {
                    printf("\n\t\t\t\t\t\tFood item with given serial number already
exists!!\n\n");
                    flag = 1;
                    break;
                }
                temp = temp->next;
            }

            if (flag == 1)
                break;

            printf("\t\t\t\t\t\tEnter food item name: ");
            scanf("%s", name);
            printf("\t\t\t\t\t\tEnter price: ");
            scanf("%f", &price);
            heada = createadmin(heada, num, name, price);
            printf("\n\t\t\t\t\t\tNew food item added to the list!!\n\n");
            break;
        case 3:
            system("cls");
            if (deleteadmin())
            {
                printf("\n\t\t\t\t\t\t\t### Updated list of food items menu ###\n");
                printf("\n\n\t\t\t\t\t\t    Serial No.       Name               Rate");

                displayList(heada);
            }
            else
                printf("\n\t\t\t\t\t\tFood item with given serial number doesn't
exist!\n\n");

            break;
```

```
        case 4:
            system("cls");
            printf("\n\t\t\t\t\t\t\t   ### Order menu ###\n");
            printf("\n\t\t\t\t\t\t    Serial No.       Name               Rate");
            displayList(heada);
            printf("\n\n\t\t\t\t\t\t||Press any key to continue||");
            getch();
            printf("\n\n");
            break;

        default:
            printf("\n\t\t\t\t\t\tWrong Input !! PLease choose valid option\n");
            break;
        }
    }
}

void customer()
{
    int flag = 0, j = 1;
    char ch;
    printf("\n\t\t\t\t\t   ----------------------------------------------\n");
    printf("\t\t\t\t\t\t    CUSTOMER SECTION\n");
    printf("\t\t\t\t\t   ----------------------------------------------\n");
    while (1)
    {
        customermenu();
        int opt;
        scanf("%d", &opt);

        if (opt == 5)
            break;

        switch (opt)
        {
        case 1:
            system("cls");
            displayList(heada);
            printf("\n\t\t\t\t\t\tEnter number corresponding to the item you want to
order: ");
            int n;
            scanf("%d", &n);
            printf("\t\t\t\t\t\tEnter quantity: ");
            int quantity;
            scanf("%d", &quantity);
            headc = createcustomer(headc, n, quantity);
            break;
        case 2:
            system("cls");
            printf("\n\t\t\t\t\t\t  ### List of ordered items ###\n");
            displayList(headc);
            break;
        case 3:
            system("cls");
```

```
            if (deletecustomer())
            {
                printf("\n\t\t\t\t\t\t### Updated list of your ordered food items ###\n");
                printf("\n\n\t\t\t\t\t\t    Serial No.       Name     Quantity
 Rate");

                displayList(headc);
            }
            else
                printf("\n\t\t\t\t\t\tFood item with given serial number doesn't
exist!!\n");
            break;
        case 4:
            system("cls");

            calculatetotsales();
            printf("\n\t\t\t\t\t\t\t  ### Final Bill ###\n");
            displaybill();
            headc = deleteList(headc);
            printf("\n\t\t\t\t\t\tPress any key to return to main menu:\n\t\t\t\t\t\t");
            fflush(stdin);
            ch = fgetc(stdin);
            flag = 1;
            break;

        default:
            printf("\n\t\t\t\t\t\tWrong Input !! PLease choose valid option\n");
            break;
        }
        if (flag == 1)
            break;
    }
}

void mainnenu()
{

printf("\n\t\t\t\t\t\t_____________________________________________________________
____\n");
    printf("\t\t\t\t\t\t1. ADMIN SECTION(Lodin credentials required)--> \n");
    printf("\t\t\t\t\t\t2. CUSTOMER SECTION--> \n");
    printf("\t\t\t\t\t\t3. Exit--> \n\n");
    printf("\t\t\t\t\t\tEnter Your Choice --->");
}

int main()
{

printf("\n\n\n\n\t\t\t\t@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@\n");
    printf("\t\t\t\t@@
_____________________________________________________________________________
@@\n");
```

```
    printf("\t\t\t\t@@|
                |@@\n");
    printf("\t\t\t\t@@|
                |@@\n");
    printf("\t\t\t\t@@|
                |@@\n");
    printf("\t\t\t\t@@|
                |@@\n");
    printf("\t\t\t\t@@|
                |@@\n");
    printf("\t\t\t\t@@|
                |@@\n");
    printf("\t\t\t\t@@|                                    WELCOME TO
                    |@@\n");
    printf("\t\t\t\t@@|
                    |@@\n");
    printf("\t\t\t\t@@|                  \xB3\xB2=\xB2=\xB2-\xB3 RESTAURANT MANAGEMENT
SYSTEM  \xB3\xB2=\xB2=\xB2-\xB3                      |@@\n");
    printf("\t\t\t\t@@|
                    |@@\n");
    printf("\t\t\t\t@@|
                    |@@\n");
    printf("\t\t\t\t@@|
                    |@@\n");
    printf("\t\t\t\t@@|
                    |@@\n");
    printf("\t\t\t\t@@|                                       -Brought To You by DEVVART
  2k20/IT/41       |@@\n");
    printf("\t\t\t\t@@|                                                    and
DHARAMVEER  2k20/IT/42       |@@\n");

printf("\t\t\t\t@@|__________________________________________________________________________________________
________________|@@\n");

printf("\t\t\t\t@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@\n\n\n\n\t\t\t\t\t");

    printf("\t\n\n\t\t\t\t\t\t\t CONTINUE[Press any key]  ");
    getch();
    printf("\n");
    heada = createadmin(heada, 1, "Hot and Sour Soup", 100);
    heada = createadmin(heada, 2, "Manchow Soup", 200);
    heada = createadmin(heada, 3, "Manchurian Noodles", 150);
    heada = createadmin(heada, 4, "Fried Rice", 180);
    heada = createadmin(heada, 5, "Hakka Noodles", 80);
    heada = createadmin(heada, 5, "Afghan Jalebi", 400);

    while (1)
    {
        mainnenu();
        int choice;
        scanf("%d", &choice);

        if (choice == 3)
```

```
        {
            printf("\n\n\t\t\t\t\t\t\t**********Thank you!!**********\n");
            break;
        }

        switch (choice)
        {
        case 1:
        {
            int k = 3;

        B:

            printf("\n\n\t\t\t\t\t  ****************************************");
            printf("\n\t\t\t\t\t\t  Enter User name and Password \n");
            printf("\t\t\t\t\t  ****************************************\n");
            while (k >= 1)
            {
                int c = password();
                if (c == 1)
                {
                    printf("\n\t\t\t\t\t\t ******LOG IN SUCCESSFUL*******");
                    Sleep(2000);
                    break;
                }
                else
                    printf("\n\n\t\t Wrong Password Or User Name \n\n\t\t You Can try Only
%d times more", k - 1);
                k--;
                if (k < 1)
                {
                    for (int i = 59; i >= 0; i--)
                    {
                        system("cls");
                        printf("\n\n\n\n\n\n\n\n\t\t\t\t\tYOU ARE BLOCKED FOR 1
MINUTE!!");
                        printf("\n\n\n\n\n\n\t\t\t\t\tTRY AFTER %d SECONDS.....", i);
                        Sleep(1000);
                    }
                    k = 3;
                    goto B;
                }
            }
            admin();
            break;
        }
        case 2:
            system("cls");
            customer();
            break;
        case 3:

            break;
```

```
        default:
            printf("\n\t\t\t\t\t\tWrong Input !! PLease choose valid option\n");
            break;
        }
    }
}
```