NAME: Vyde Siddhartha

PROJECT TYPE: Self Project

NAME of the Project: Restaurant Management

INSTITUTE: IIT Kharagpur

RESTAURANT MANAGEMENT SYSTEM

INTRODUCTION

Restaurant management is the profession of managing a restaurant. It includes the major function of planning, organizing, staffing, directing, developing an attitude in food and beverage control systems and to efficiently and effectively plan menus at profitable prices, taking into thought constraints, preparation and other variables affecting food and beverage outlets.

The Restaurant Management Software program is a comprehensive restaurant management tool designed for foodservice management of all types. It is easy to be told and simple to use. This system processes group action and stores the ensuing knowledge. Reports are going to be generated from this knowledge that facilitate the manager to form applicable business choices for the building. For example, knowing the number of customers for a particular time interval, the manager can decide whether more waiters and chefs are required. Restaurant Software Systems are essential to the successful operation of most foodservice establishments because they allow the business to track transactions in real-time

In short, the program simply is viable and may be easily accessed by a user. It is an excellent software system for the long run generation because it saves time and reduces the work of the owner of the building and also the waiter too. It will additionally facilitate the homeowners to handle their customers in a very higher and a snug approach.

Modules:

We divided the project into several modules for the better understanding .The modules help us Handling the errors and access each class properly.Menu Management is done by following features.

- **Add menu items:**

In this we Add Items in a menu by assigning a unique number to each item. When we add any item we give description as category, full name and price for each item. The access of the item is given to the Owner of the restaurant.

- **View Full Menu :**

As the name indicates in this we display the full information f all the items in a menu. Only outlook of all the items is displayed, no editing is performed.

- **Selecting a category :**

In this we can select different categories in our menu i.e, Indian Cuisine, Chinese Cuisine, Chicken and Mutton, Continental, Beverages, Desserts and Soups.

- **Choosing a item:**

In this we can select multiple items from different categories according to the choice and respectively having a look on the price of it.

- **Deleting:**

An item in this category after selecting the items from different categories we can delete an item if we want and accordingly these bill is managed at the end.

- **Billing:**

In this we can see the resultant bill of the items selected by the user and accordingly the user can verify the items and the price and can be sure of no mistake.

Concepts used in: -

1. Linked list
2. Queues

Advantages: -

• It increases operational efficiency.

• It is designed to help you cost your recipes and track inventory saving your Money and Time and maximizing profit.

• It helps the restaurant manager to manage the restaurant more effectively and efficiently by computerizing Meal Ordering, Billing, and Sales Management.

• Accounting.

• It increases the security.

• It avoids paper work.

• It is Simple to learn and easy to use.

• It is portable

ALGORITHM:

Algorithm for main():

```
Step 1: Start.
Step 3: Display “Welcome to Departmental Store Management System”.
Step 4: Call function mainmenu().
Step 5: Stop.
Orders management
```

Algorithm for item inventory():

```
Step 1: Start.
Step 2: Display “item inventory".
Step 3: Display “Enter the choices: (L) to login, (C) to create Manager/Cashier, (E) to exit".
Step 4: Input choice1.
Step 5:
        5.1 : if choice1=76 or choice1=108
                        Call function Verification().
                else
                if choice1=67 or choice1=99
                               Display “Enter (M) to create Manager ID.")
                               Display "Enter (C) to create Cashier ID."
                               Display "Enter (R) to return to main menu:"
        5.2 : Input choice2.
        5.3 : if choice2=77 or choice2=109
                        Call function CreateManager().
                else
                if choice2=67 or choice2=99
                        Call function Createemployee().
                else
                if choice2=82 or  choice2=114
                        Call function mainmenu().
                else
                        Display "Invalid Selection! Enter again:"
                        Goto step 5.2.
Step 6: if choice1=69 or choice 1=101
                exit.
         else
                Display "Invalid Selection! Enter again:"
                Goto step 4.
```

```
Step 7: Stop.
```

Algorithm for items Entry():

```
Step 1: Start.
Step 2: Declare file pointer *f.
Step 3: Declare variable ques.
Step 4: Open file “items.txt” in append mode and assign to pointer f.
Step 5: Display “Enter the item code to be added:”
Step 6: Input a.itemcode.
Step 7: Display “Enter the item name:”
Step 8: Input a.itemname.
Step 9: Display “Enter the rate for unit quantity:”
Step 10:Input a.rate.
Step 11:Write structure a in f.
Step 12:Display “Enter the 'esc' key to exit or any other key to continue:”
Step 13:Input ques.
Step 14:if (ques=27)
                Goto step 15.
        else
                Goto step 5.
Step 15:Close file f.
Step 16:Call function ManagerMenu().
Step 17:Stop.
```

Algorithm for items List Display():

```
Step 1: Start.
Step 2: Declare file pointer *f.
Step 3: Open file “items.txt” in read mode and assign it to f.
Step 4: Display “Item code Item name Cost”. Step 5: Display
“------------------------------------------ “. Step 6: Print
a.itemcode, a.itemname, a.rate.
Step 7: if (f=EOF)
                Goto Step 8.
        else
                Goto Step 6.
Step 8: Close file f.
Step 9: Call function ManagerMenu().
```

Algorithm for Edit (erasing incorrect record)items():

```
Step 1: Start.
Step 2: Declare file pointer *f.
Step 3: Declare variable itemcode[50], size.
Step 4: Open file “goods.txt” in read mode and assign to file pointer f.
Step 5: Display “Enter the item code of the product:” Step 6: Input
itemcode.
Step 7: Read f.
Step 8: if (f=EOF)
                Goto Step 11.
         else
         if (a.itemcode=itemcode)
                Display “Enter the new item code:”
         Input a.itemcode,
                Display “Enter the new item name:”
         Input a.itemname,
                Display “Enter the rate for unit quantity:”
                Input a.itemname.
Step 9: fseek(f,-size,SEEK_CUR)
Step 10:Write structure a.
Step 11: Close f.
Step 12:Call function ManagerMenu().
Step 13:Stop.
```

Algorithm for removes the items():

```
Step 1: Start.
Step 2: Declare file pointer *f, *temp.
Step 3: Declare variable itemcode[50], size.
Step 4: Open file “items.txt” in read mode and assign to file pointer f. Step 5:
Open file “tempitems.txt” in write mode and assign to file pointer temp. Step 6:
Display “Enter the item code of the product to be deleted:”
Step 7: Input itemcode.
Step 8: Read f.
         if (f=EOF)
                Goto Step 9.
         else
        if (itemcode=a.itemcode)
                Goto Step 8.
         else
                Write a in temp.
Step 9: Remove file “items.txt”.
Step 10:Rename “tempitems.txt” to “goods.txt”.
Step 11:Close f.
Step 12:Close temp.
Step 13: Call function ManagerMenu().
Step 14:Stop.
```

Algorithm for Billing():

```
Step 1: Start.
Step 2: Declare structure goods b[50] and variables itemcode[50], i, j, sum, quantity[200],
        cost[200], tender, change.
Step 3: Declare file pointer *f, *f1.
Step 4: Open file “goods.txt” in read mode and assign to f.
Step 5: Open file “keeprecords.dat” in append mode and assign to f1.
Step 6: Initialize i=0.
Step 7: Display “Enter the item codes and enter any key when done.”
Step 8: Display “Enter the item codes:”
Step 9: Input itemcode.
Step 10:if (strlen(itemcode)=6)
                Goto Step 11.
        else
                Goto Step 16.
Step 11:Read f.
Step 12:if (a.itemcode=itemcode)
                Print a.itemname, a.rate,
        b[i].itemcode=a.itemcode,
        b[i].itemname=a.itemname,
        b[i].rate=a.rate,
        Display "Enter the quantity:"
                Input quantity[i],
                cost[i]=quantity[i]*b[i].rate,
        sum=sum+cost[i],
                c.itemcode=b[i].itemcode,
        c.itemname=b[i].itemname,
        c.rate=b[i].rate,
        c.quantity=quantity[i],
        c.cost=cost[i].
Step 13:Write c in f1.
Step 14:i++,
Step 15:Goto Step 8.
Step 16:Close f and f1.
Step 17:Display “Item code Item name Cost Qty Amount”. Step
18:Display “ ------------------------------------------ ”.
Step 19:Initialize j=0.
Step 20:if (j<i)
                Print b[j].itemcode, b[j].itemname, b[j].rate, quantity[j],
                cost[j]. j=j+1,
                Goto step 18.
Step 21:Print sum.
Step 22:Display “Tender:”
Step 23:Input tender.
Step 24:change=tender-sum.
Step 25:Print change.
```

EXISTING ALGORITHM:

```
1.first=new node;{create the 1st node of the list pointed by first};
2.Read(Data(first));
3.NEXT(First)=NULL;
4.Far a First; [point Far to the First]
5. For I=1 to N-1 repeat steps 6 to 10
6.X=new node;
7.Read(Data(X))
8.NEXT(X)=NULL;
9.NEXT(Far)=X; {connect the nodes}
10.Far=X;[shift the pointer to the last node of the list]
[end of For Loop]
11.END
```

ADD AN ITEM TO LIST

```
1. Step 1. Create a new node and assign the address to any node say ptr.
2. Step 2. OVERFLOW,IF(PTR = NULL)
3. write : OVERFLOW and EXIT.
4. Step 3. ASSIGN INFO[PTR] = ITEM
5. Step 4. IF(START = NULL)
6. ASSIGN NEXT[PTR] = NULL
7. ELSE
8. ASSIGN NEXT[PTR] = START
9. Step 5. ASSIGN START = PTR
10. Step 6. EXIT
```

DELETE AN ITEM AT FIRST

```
1. DELETE AT BEG(INFO,NEXT,START)
2.
3. 1.IF(START=NULL)
4.
5. 2.ASSIGN PTR = STRAT
```

```
3.ASSIGN TEMP = INFO[PTR]
4.ASSIGN START = NEXT[PTR]
5.FREE(PTR)
6.RETURN(TEMP)
```

DELETE AT LAST

```
Delete End(info,next,start)
1.if(start=NULL)
Print Underflow and Exit.
2.if(next[start]==NULL)
Set ptr =start and start=NULL.
set temp = info[ptr].
else cptr = start and ptr = next[start].
Repeat steps(a) and (b) while(next[ptr]!=NULL)
(a) set cptr = ptr
(b) set ptr = next[ptr]
[end while]
set next[cptr] = NULL
temp = info[ptr]
[end if]
3. free(ptr)
4. return temp
5. exit
```

CODE:

```
#include<stdlib.h>
```

```
#include<stdio.h>

#include<string.h>

struct order

    {

    char name[100];

    int month;

    int year;

    int date;

    };

struct node *create(struct node *start);

void display(struct node *start);

struct node *delete_func(struct node *start);

void enqueue_order();

void display_order();

void dequeue_order();

void check(struct order temp);

struct node

{

    int expenditure;

    char name[100];

    struct node *next;

};

struct order queue[100];

int rear=-1,front=-1;
```

```
struct node *item_start=NULL;

struct node *employee_start=NULL;

void order_manager()

{

    int choice=0;

    while(choice!=5)

    {

        printf("\n 1.Add a New Order .\n");

        printf(" 2.Display the Pending Orders\n");

        printf(" 3.Remove the Upcoming Order.\n");

        printf(" 4.Exit\n");

        printf("Enter your choice:\t");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

                enqueue_order();

                break;

            case 2:

                display_order();

                break;

            case 3:

                dequeue_order();

                break;
            case 4:
                continue;
                break;
            default:
                printf("\n");
                break;
        }
    }
}
void assign(int i,struct order temp)
{
    int j;
    for (j = rear + 1; j > i; j--)
    {
        queue[j] = queue[j - 1];
    }
    strcpy(queue[rear].name,temp.name);queue[i].year = temp.year;
    queue[i].month = temp.month;
    queue[i].date = temp.date;
    }
void enqueue_order()
{
    struct order temper;
    printf("Enter the Name of delivery Reciptant: ");
```

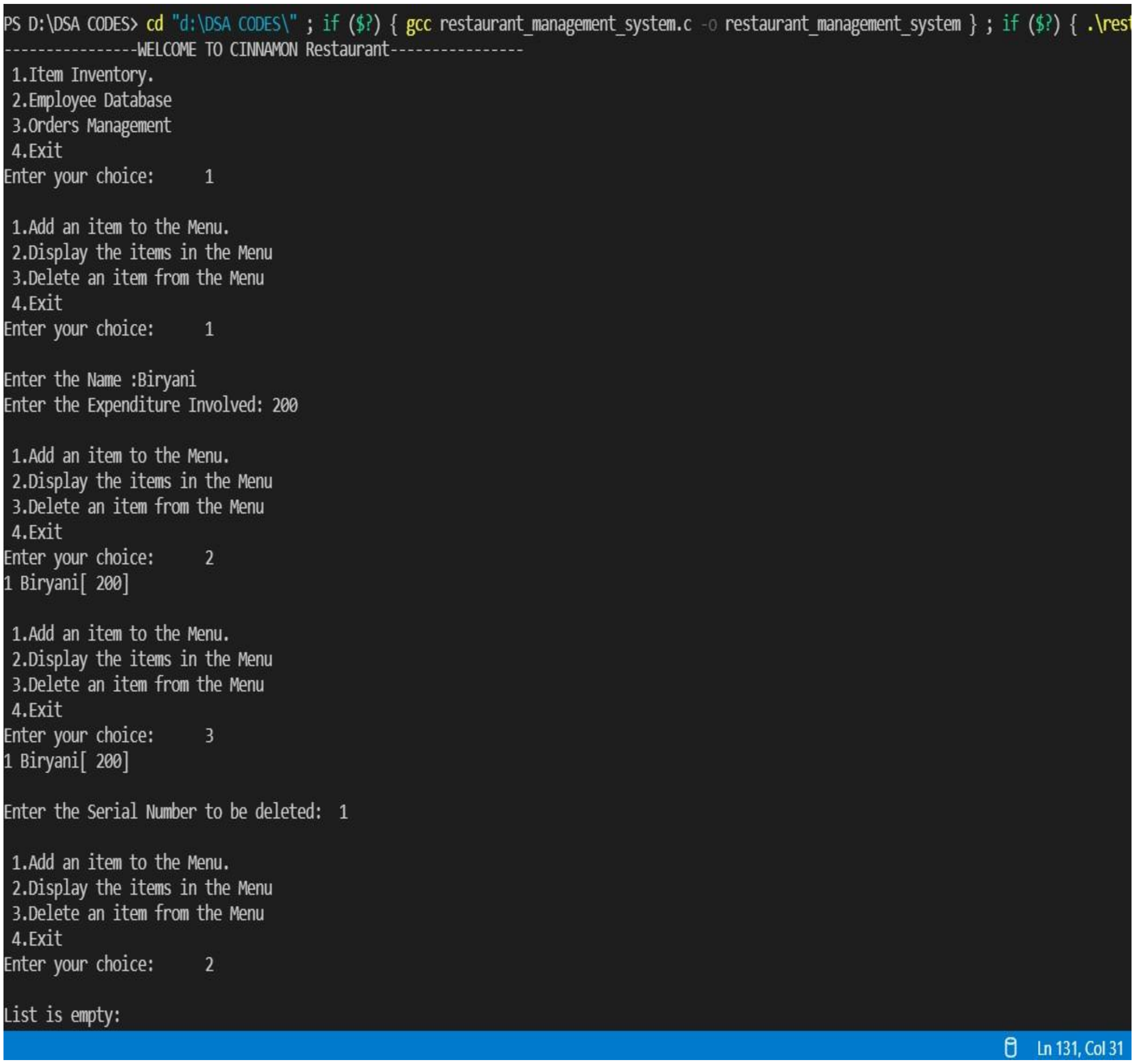
```
scanf("%s",temper.name);
printf("\nEnter the Year of delivery: ");
scanf("%d",&temper.year);
printf("Enter the month of delivery: ");
scanf("%d",&temper.month);
printf("Enter the date of delivery: ");
scanf("%d",&temper.date);
if (rear >=99)
{
   printf("\nThe Hotel takes in only 100 orders at a time.\n");
   return;
}
if ((front == -1) && (rear == -1))
{
   front++;
   rear++;
   strcpy(queue[rear].name,temper.name);
   queue[rear].year=temper.year;
   queue[rear].month=temper.month;
   queue[rear].date=temper.date;
   return;
}
else
   check(temper);
```

```
    rear++;

};

void check(struct order temp)

{

    int i,j;

    for (i = 0; i <= rear; i++)

    {

        if (temp.year < queue[i].year)

        {

            assign(i,temp);

            return;

        }

        else if( temp.year==queue[i].year)

        {

            if (temp.month < queue[i].month)

            {

            assign(i,temp);

            return;

            }

            else if(temp.month==queue[i].month)

            {

                if(temp.date<=queue[i].date)

                {

                    assign(i,temp);
                return;

            }

        }

    }

    }

    strcpy(queue[rear].name,temp.name);

    queue[i].year = temp.year;

    queue[i].month = temp.month;

    queue[i].date = temp.date;

}

void display_order()

{

    int temperory=front;

    if ((front == -1) && (rear == -1)||(front==rear && front!=0))

    {

        printf("\nNo orders Yet.");

        return;

    }

    for (temperory; temperory <= rear; temperory++)

    {

        printf("%s
=>%d.%d.%d\n",queue[temperory].name,queue[temperory].date,queue[temperory].month,qu
eue[temperory].year);

    }

}
```

```
void dequeue_order()
   {
   if((rear==-1 && front==-1) || (rear<=front))
   {
      printf("\nNo Orders Pending.\n");
   }
   else
      front++;
}
void employee_menu()
{
   int choice=0;
   while(choice!=4)
   {
      printf("\n 1.Appoint a New Staff Member .\n");
      printf(" 2.Display the Current Staff\n");
      printf(" 3.Delete a Staff Member from the DataBase\n");
      printf(" 4.Exit\n");
      printf("Enter your choice:\t");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
            employee_start=create(employee_start);
            break;
        case 2:
            display(employee_start);
            break;
        case 3:
            employee_start=delete_func(employee_start);
            break;
        case 4:
            continue;
            break;
        default:
            printf("\n Wrong Choice:\n");
            break;
      }
   }
}
void item_menu()
{
   int choice=0;
   while(choice!=4)
   {
      printf("\n 1.Add an item to the Menu.\n");
      printf(" 2.Display the items in the Menu\n");
      printf(" 3.Delete an item from the Menu\n");
```

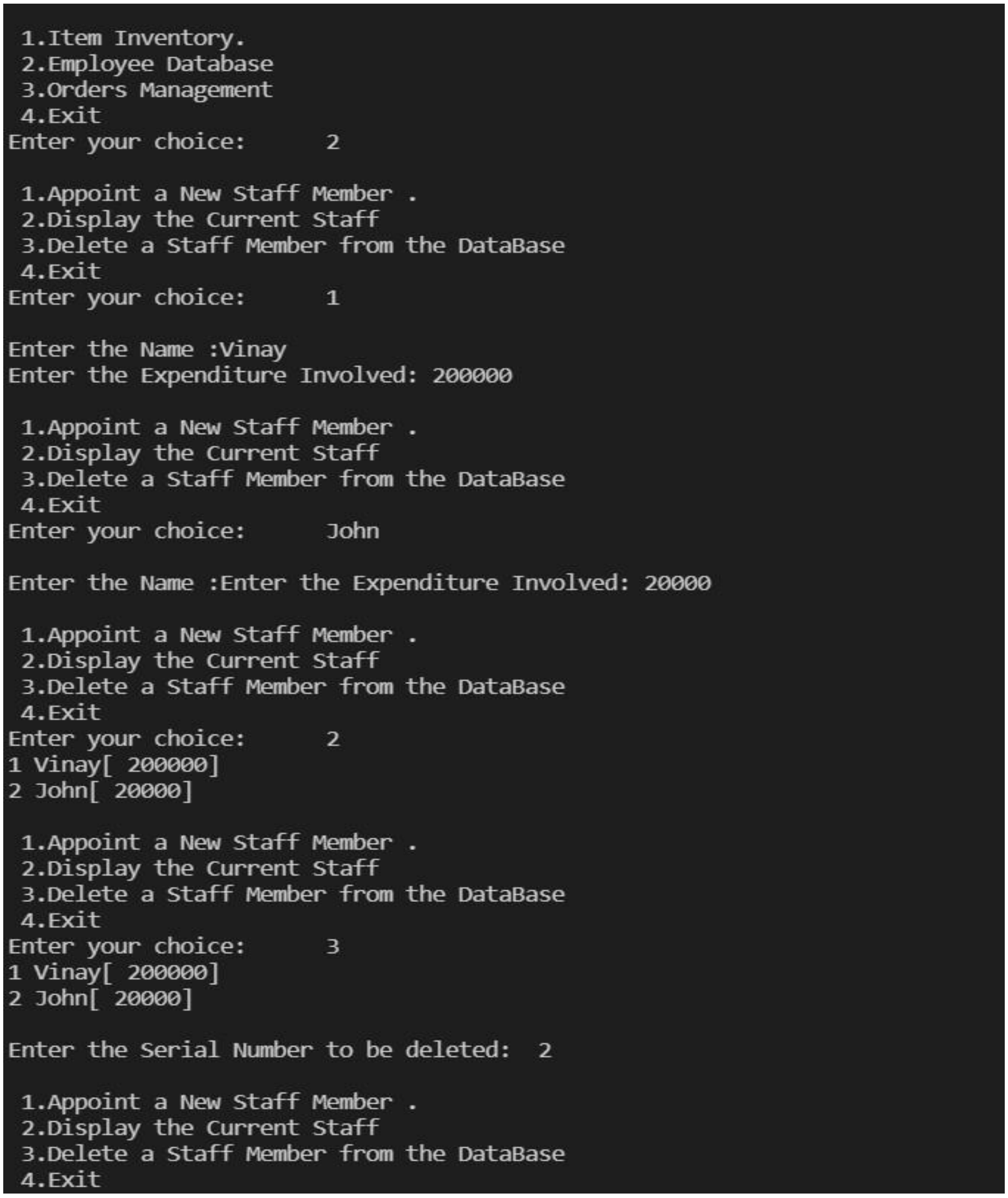
```
        printf(" 4.Exit\n");
        printf("Enter your choice:\t");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                item_start=create(item_start);
                break;
            case 2:
                display(item_start);
                break;
            case 3:
                item_start=delete_func(item_start);
                break;
            case 4:
                continue;
                break;
            default:
                printf("\n Wrong Choice:\n");
                break;
        }
    }
}
int main()
```

```
{
    printf("----------------WELCOME TO CINNAMON Restaurant ------------- ");
    int choice=0;
    while(1)
    {
        printf("\n 1.Item Inventory.\n");
        printf(" 2.Employee Database\n");
        printf(" 3.Orders Management\n");
        printf(" 4.Exit\n");
        printf("Enter your choice:\t");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                item_menu();
                break;
            case 2:
                employee_menu();
                break;
            case 3:
                order_manager();
                break;
            case 4:
                exit(0);
            break;

          default:

            printf("\n Wrong Choice:\n");

            break;

        }

    }

    return 0;

}

struct node *create(struct node *start)

{

    struct node *temp,*ptr;

    int count=0;

    temp=(struct node *)malloc(sizeof(struct node));

    if(temp==NULL)

    {

        printf("\nOut of Memory Space:\n");

        exit(0);

    }

    printf("\nEnter the Name :");

    scanf("%s", temp->name);

    printf("Enter the Expenditure Involved: ");

    scanf("%d",&temp->expenditure);

    temp->next=NULL;

    if(start==NULL)
    {
        start=temp;
    }
    else
    {
        ptr=start;
        while(ptr->next!=NULL){
        ptr=ptr->next;
        }
        ptr->next=temp;
    }
    return(start);
}
void display(struct node *start)
{
    struct node *ptr;
    int count=1;
    if(start==NULL)
    {
        printf("\nList is empty:\n");
        return;
    }
    else
    {
        ptr=start;
        while(ptr!=NULL)
        {
            printf("%d %s[ %d]\n",count,ptr->name,ptr->expenditure );
            count++;
            ptr=ptr->next ;
        }
    }
}
struct node *delete_func(struct node *start)
{
    int i,pos;
    struct node *temp,*ptr;
    if(start==NULL)
    {
        printf("\nThe List is Empty:\n");
    }
    else
    {
        display(start);
        printf("\nEnter the Serial Number to be deleted:\t");
        scanf("%d",&pos);
        if(pos==1)
        {
            ptr=start;

            start=start->next ;

            free(ptr);

        }

        else

        {

            ptr=start;

            for(i=2;i<=pos;i++)

            {

                temp=ptr;

                ptr=ptr->next ;

                if(ptr==NULL)

                {

                    printf("\nSerial Number not Found:\n");

                    return(start);

                }

            }

            temp->next =ptr->next ;

            free(ptr);

        }

    }

    return(start);

}
```

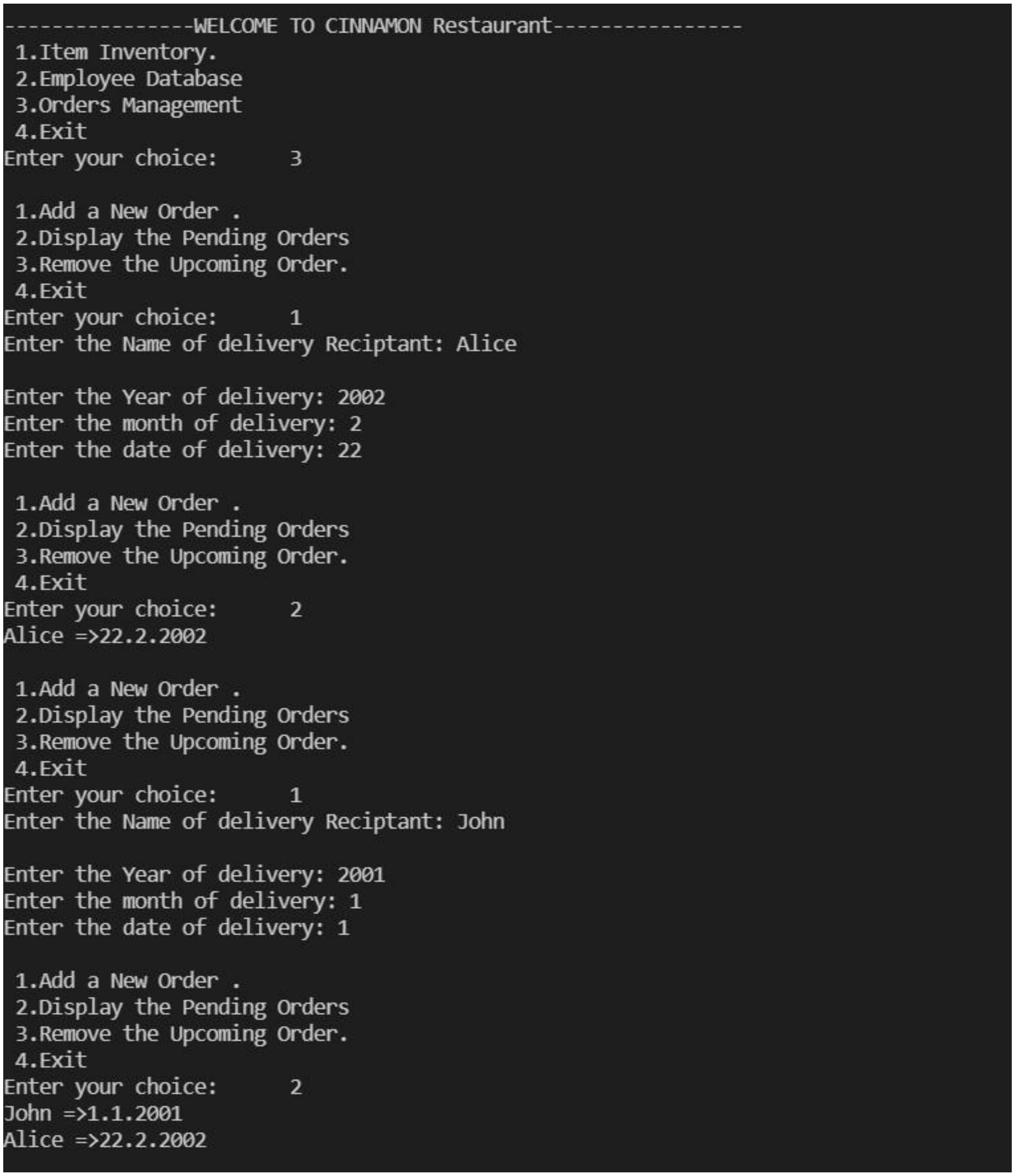


```
PS D:\DSA CODES> cd "d:\DSA CODES\" ; if ($?) { gcc restaurant_management_system.c -o restaurant_management_system } ; if ($?) { .\rest
---------------WELCOME TO CINNAMON Restaurant---------------
 1.Item Inventory.
 2.Employee Database
 3.Orders Management
 4.Exit
Enter your choice:      1

 1.Add an item to the Menu.
 2.Display the items in the Menu
 3.Delete an item from the Menu
 4.Exit
Enter your choice:      1

Enter the Name :Biryani
Enter the Expenditure Involved: 200

 1.Add an item to the Menu.
 2.Display the items in the Menu
 3.Delete an item from the Menu
 4.Exit
Enter your choice:      2
1 Biryani[ 200]

 1.Add an item to the Menu.
 2.Display the items in the Menu
 3.Delete an item from the Menu
 4.Exit
Enter your choice:      3
1 Biryani[ 200]

Enter the Serial Number to be deleted:  1

 1.Add an item to the Menu.
 2.Display the items in the Menu
 3.Delete an item from the Menu
 4.Exit
Enter your choice:      2

List is empty:
```

Ln 131, Col 31



```
 1.Item Inventory.
 2.Employee Database
 3.Orders Management
 4.Exit
Enter your choice:      2

 1.Appoint a New Staff Member .
 2.Display the Current Staff
 3.Delete a Staff Member from the DataBase
 4.Exit
Enter your choice:      1

Enter the Name :Vinay
Enter the Expenditure Involved: 200000

 1.Appoint a New Staff Member .
 2.Display the Current Staff
 3.Delete a Staff Member from the DataBase
 4.Exit
Enter your choice:      John

Enter the Name :Enter the Expenditure Involved: 20000

 1.Appoint a New Staff Member .
 2.Display the Current Staff
 3.Delete a Staff Member from the DataBase
 4.Exit
Enter your choice:      2
1 Vinay[ 200000]
2 John[ 20000]

 1.Appoint a New Staff Member .
 2.Display the Current Staff
 3.Delete a Staff Member from the DataBase
 4.Exit
Enter your choice:      3
1 Vinay[ 200000]
2 John[ 20000]

Enter the Serial Number to be deleted:  2

 1.Appoint a New Staff Member .
 2.Display the Current Staff
 3.Delete a Staff Member from the DataBase
 4.Exit
```



```
----------------WELCOME TO CINNAMON Restaurant----------------
 1.Item Inventory.
 2.Employee Database
 3.Orders Management
 4.Exit
Enter your choice:      3

 1.Add a New Order .
 2.Display the Pending Orders
 3.Remove the Upcoming Order.
 4.Exit
Enter your choice:      1
Enter the Name of delivery Reciptant: Alice

Enter the Year of delivery: 2002
Enter the month of delivery: 2
Enter the date of delivery: 22

 1.Add a New Order .
 2.Display the Pending Orders
 3.Remove the Upcoming Order.
 4.Exit
Enter your choice:      2
Alice =>22.2.2002

 1.Add a New Order .
 2.Display the Pending Orders
 3.Remove the Upcoming Order.
 4.Exit
Enter your choice:      1
Enter the Name of delivery Reciptant: John

Enter the Year of delivery: 2001
Enter the month of delivery: 1
Enter the date of delivery: 1

 1.Add a New Order .
 2.Display the Pending Orders
 3.Remove the Upcoming Order.
 4.Exit
Enter your choice:      2
John =>1.1.2001
Alice =>22.2.2002
```

CONCLUSION: -

This project entitled “RESTAURANT MANAGEMENT SYSTEM” has very good scope in the practical world. The main advantage of this project is that its simplicity attracts tons of users. It is simply go by a novice user. This software can be used in any kind of restaurant .This Management System helps the building manager to manage the restaurant a lot of effectively and expeditiously by computerizing meal ordering, billing and inventory control.
Also, there will be speedy and secured authentication procedure for the maintenance of records. Data entry is fast and simple. Therefore, this computer code can be used in replacing the obsolete manual methodology of maintaining secure records. The work plan also includes the elaborate options of the technology utilized in the project shaping the front end and back end.