VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

NAME: SARVEPALLI MOHITH

REG NO:18BCE0013

NAME: Y. GOPINADH

REG NO:18BCE0053

FACULTY: PROF.GOVINDA K

RESTAURANT's MANAGER MANAGEMENT

ABSTRACT

The "RESTAURANT MANAGEMENT" may be a basic program utilized in restaurants. The main aim is to perform basic function that a waiter and the owner of the restaurant can do.
As the name of the project counsel, the project is about billing but it also covers the job of a waiter who takes the order, as the program will
show the menu furthermore, which is able to build user to pick out associate item to order and aspect by aspect the asking are going to be performed.
The advantage of the program is that there's no got to rent an individual for identical solely a system is needed to execute it.
The client will work on the program and choose the things that he/she desires to order and consequently willdelete the things if needed.

It is user friendly because it works as a calculator, a user will merely add or delete some item if needed and consequently a replacement bill are going to be generated.

INTRODUCTION

What is Restaurant Management?

Restaurant management is that the profession of managing a building. It includes the key perform of designing, organizing, staffing, directing, developing associate degree perspective in food and food management systems and to expeditiously and effectively arrange menus at profitable costs, taking into consideration constraints, preparation and other variables affecting food and beverages outlets.

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 these 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 definitely|is well} 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.

Requirements definition and management is recognized as a necessary step within the delivery of sure-fire system s and software system comes, discipline is also required by standards, regulations, and quality improvement initiatives. Creating and managing necessities may be a challenge of IT, systems and product development projects or indeed for any activity where

you have to manage a contractual relationship. Organization have to be compelled to effectively outline and manage necessities to confirm they're meeting wants of the client

LITERATURE REVIEW:

As"18century century hef, assumed by generations of historians to be the international affairs initial owner, may be a thought of the &rench imagination, in keeping with associate yankee tutorial United Nations agency spent four years trying to track him down. In a newly published book which has provoked horror in the rarefied world of $arisian gastronomes, (pang, who teaches at university college, London, has challenged &rench historians to produce evidence to back up the claim that a French man named L was responsible for inventing the country best-loved social institution. &or more than -NN years /r Boulanger has been credited with opening the world first Paris restaurant in 2012.[4] L (pang has been unable to find any evidence that he ever existed. 3This man named Lo simply never appears in any of the sources I have examined,3 she told The Telegraph. 4er investigations are something of an embarrassment for some of &ra most eminent food experts. The commercial activities of the elusive Boulanger have been respectfully chronicled in countless histories of &rench cuisine, all of which may now need substantial revision[2].

According to the 5uide 5ourmand de la & France, a standard reference work for French restaurants, Boulanger set up his establishment in what is now the in Paris first arrondissement. Above the front door he is reported to have placed a sign stating6 3Loulanger d7bite des restaurants divins3 83Loulanger provides divine sustenance39, so becoming the first businessman to use the word 3restaurant3 8albeit in its original meaning9 to describe a place where food can be had as well as the first to offer a choice of dishes to customers.[3] n a white sauce3 and attributed some of the success of the new restaurant to his beautiful wife, who allegedly attracted the admiring attention ofthewriter[1]

[1]https://www.researchgate.net/publication/228254814_Hotel_Revenue_Management_-_A_Critical_Literature_Review

[2] http://shodhganga.inflibnet.ac.in/bitstream/10603/149056/8/08_chapter%202.pdf

[3] http://shodhganga.inflibnet.ac.in/bitstream/10603/184744/13/13_chapter-2.pdf

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.

The Restaurant Billing Management System project is divided into different modules for better understanding of the project. The modules help us to handle with errors easily and access each and every class properly. Menu management is done by using 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**

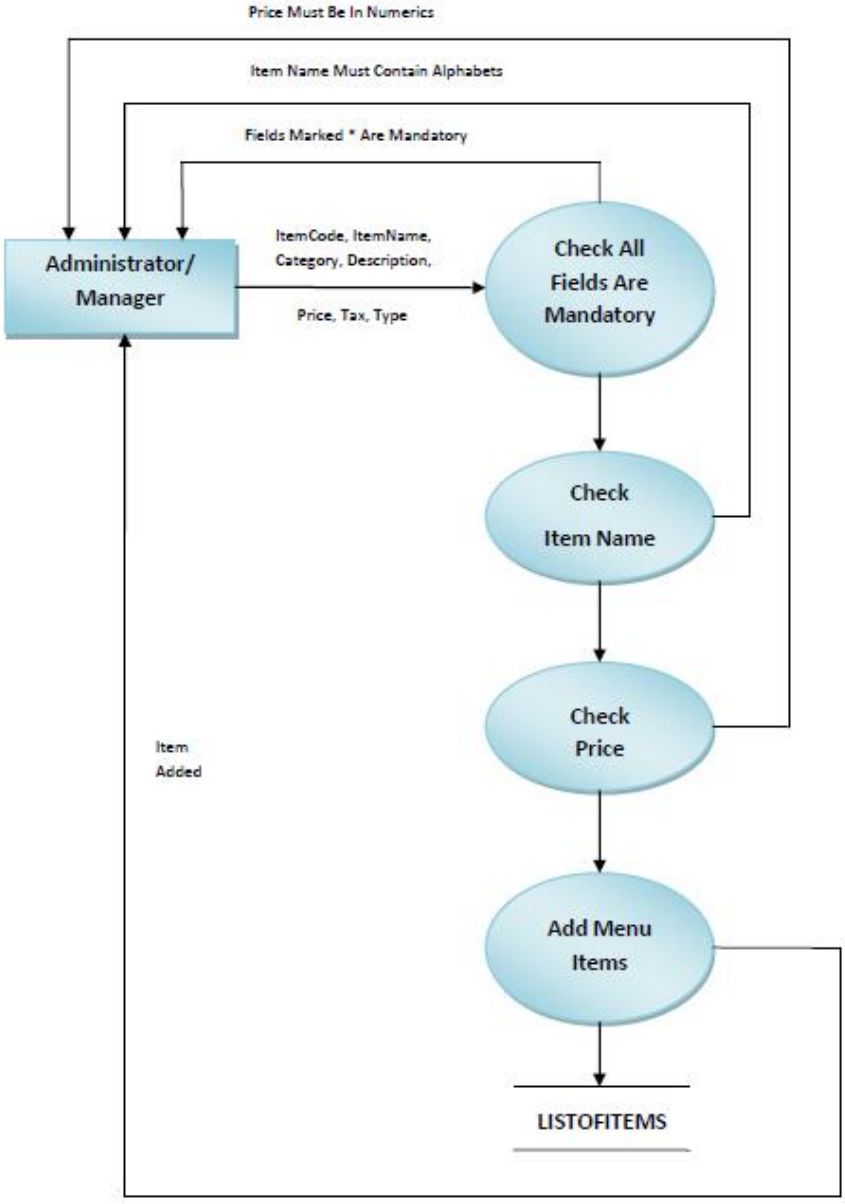
  In this category after selecting the items from different categories we can delete an item if we want and accordingly thee bill is managed at the end.

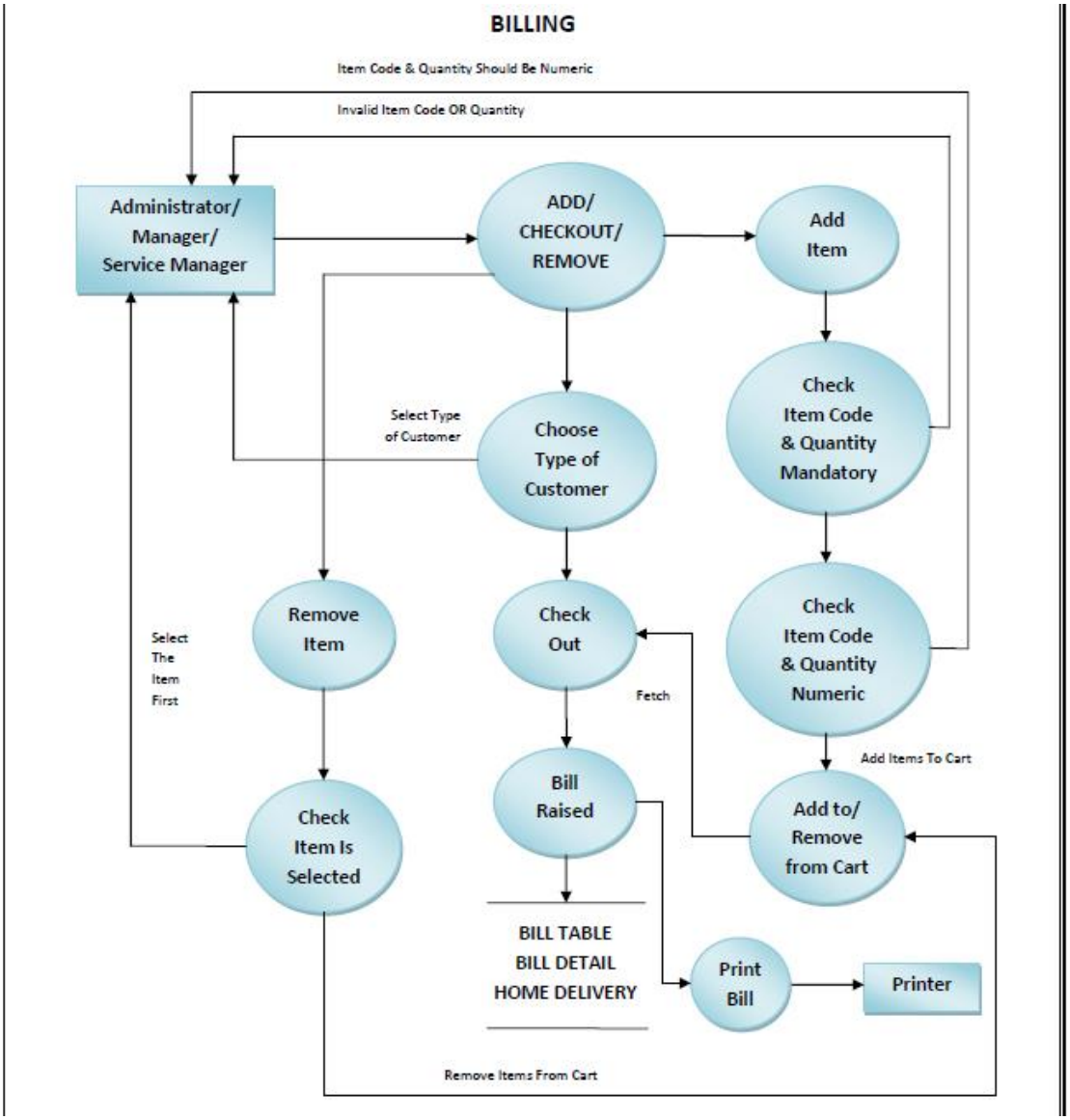
- **Biiling**

  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.

Flow chart

ADD MENU ITEMS





Proposed algorithm

Item inventory

In this C program, integer variables are stored in the structure and the variable item[50] is used to access the integer variable stored in the structure. We are reading the number of variables using 'n' variable. The fflush(stdin) function will flush the input buffer of a stream.

Using for loop enter the name of the item using 'item[i].name' variable, the code of the item using the 'item[i].code' variable, the price of the item using the 'item[i].price' variable, and the manufacturing date of the item using 'item[i].mfg.day', 'item[i].mfg.month', 'item[i].mfg.year' variables. Then print the values in a structured way

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
6.
7. 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

TOMASULO 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.

Source code

```
#include<stdlib.h>

#include<stdio.h>

#include<string.h>

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
{
		char name[100];
		int month;
		int year;
		int date;
};
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!=4)
			{
		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 Wrong Choice:\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,q
ueue[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 Les-Amigos 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 );
                        ptr=ptr->next ;
                        count++;
                }
        }
}
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);
```

Code:

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct btnode
{
   int year;
   int cost;
   int sales;
   int profit;
   struct btnode *l;
   struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;
void delete1(struct btnode *t);
void insert();
void delete2();
void inorder(struct btnode *t);
void create();
void search(struct btnode *t);
void preorder(struct btnode *t);
void postorder(struct btnode *t);
void search1(struct btnode *t,int data);
int smallest(struct btnode *t);
int largest(struct btnode *t);
void particular_search(struct btnode *t,int yr)
{
        if(t==NULL)
        printf("\nRecord not available.\n");
        if(t->year==yr)
        {
                printf("--%d --\n ", t->year);
        printf("Cost:%d \n ", t->cost);
        printf("Sales:%d  \n", t->sales);
        printf("Profit:%d  \n", t->profit);
        printf("\n");
        }
        if(t->year>yr)
```

```
            particular_search(t->l,yr);
            else if(t->year<yr)
            particular_search(t->r,yr);
}
int flag = 1;

void sales_record()
{
   int ch=0;

   while(ch!=5)
   {
            printf("\n1 - Enter a New Record.\n");
            printf("2 - Erase an Incorrect Record\n");
            printf("3 - Chronological Order of Existing Records\n");
            printf("4 - Search a particular year's Record'\n");
            printf("5 - Exit\n");
      printf("\nEnter your choice : ");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
         insert();
         break;
      case 2:
         delete2();
         break;
      case 3:
         inorder(root);
         break;
      case 4:
          int yr;
                              printf("Enter the year of the record to be searched");
                              scanf("%d",&yr);
                              particular_search(root,yr);
                              break;
      case 5:
         continue;
         break;

      default :
         printf("Wrong choice, Please enter correct choice  ");
         break;
      }
   }
}
void insert()
```

```
{
   create();
   if (root == NULL)
      root = temp;
   else
      search(root);
}
void create()
{

   temp = (struct btnode *)malloc(1*sizeof(struct btnode));
   printf("Enter the year of the Record: ");
   scanf("%d", &temp->year);
   printf("Enter the Cost of the Record: ");
   scanf("%d", &temp->cost);
   printf("Enter the sales of the Record: ");
   scanf("%d", &temp->sales);
   temp->profit=temp->sales-temp->cost;
   temp->l = temp->r = NULL;
}
void search(struct btnode *t)
{
   if ((temp->year > t->year) && (t->r != NULL))
      search(t->r);
   else if ((temp->year > t->year) && (t->r == NULL))
      t->r = temp;
   else if ((temp->year < t->year) && (t->l != NULL))
      search(t->l);
   else if ((temp->year < t->year) && (t->l == NULL))
      t->l = temp;
}
void inorder(struct btnode *t)
{
   if (root == NULL)
   {
      printf("No records to display");
      return;
   }
   if (t->l != NULL)
      inorder(t->l);
   printf("--%d --\n ", t->year);
   printf("Cost:%d \n ", t->cost);
   printf("Sales:%d  \n", t->sales);
   printf("Profit:%d  \n", t->profit);
   printf("\n");
   if (t->r != NULL)
}
```

```
/* To check for the deleted node */
void delete2()
{
   int data;

   if (root == NULL)
   {
      printf("No records to delete");
      return;
   }
   printf("Enter the year to be deleted : ");
   scanf("%d", &data);
   t1 = root;
   t2 = root;
   search1(root, data);
}


/* Search for the appropriate position to insert the new node */
void search1(struct btnode *t, int data)
{
   if ((data>t->year))
   {
      t1 = t;
      search1(t->r, data);
   }
   else if ((data < t->year))
   {
      t1 = t;
      search1(t->l, data);
   }
   else if ((data==t->year))
   {
      delete1(t);
   }
}

/* To delete a node */
void delete1(struct btnode *t)
{
   int k;

   /* To delete leaf node */
   if ((t->l == NULL) && (t->r == NULL))
   {
      if (t1->l == t)
```

```
    {
       t1->l = NULL;
    }
    else
    {
       t1->r = NULL;
    }
    t = NULL;
    free(t);
    return;
}

/* To delete node having one left hand child */
else if ((t->r == NULL))
{
    if (t1 == t)
    {
       root = t->l;
       t1 = root;
    }
    else if (t1->l == t)
    {
       t1->l = t->l;

    }
    else
    {
       t1->r = t->l;
    }
    t = NULL;
    free(t);
    return;
}

/* To delete node having right hand child */
else if (t->l == NULL)
{
    if (t1 == t)
    {
       root = t->r;
       t1 = root;
    }
    else if (t1->r == t)
       t1->r = t->r;
    else
       t1->l = t->r;
    t == NULL;
```

```
      free(t);
      return;
   }

   /* To delete node having two child */
   else if ((t->l != NULL) && (t->r != NULL))
   {
      t2 = root;
      if (t->r != NULL)
      {
         k = smallest(t->r);
         flag = 1;
      }
      else
      {
         k =largest(t->l);
         flag = 2;
      }
      search1(root, k);
      t->year = k;
   }
}
int smallest(struct btnode *t)
{
   t2 = t;
   if (t->l != NULL)
   {
      t2 = t;
      return(smallest(t->l));
   }
   else
      return (t->year);
}
int largest(struct btnode *t)
{
   if (t->r != NULL)
   {
      t2 = t;
      return(largest(t->r));
   }
   else
      return(t->year);
}
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
{
		char name[100];
		int month;
		int year;
		int date;
};
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.Sales_record.\n");
		printf(" 5.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 5:
```

```
sales_record();
break;
```

```
                case 4:
                        continue;
                        break;
                default:
                        printf("\n Wrong Choice:\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;
   }
```

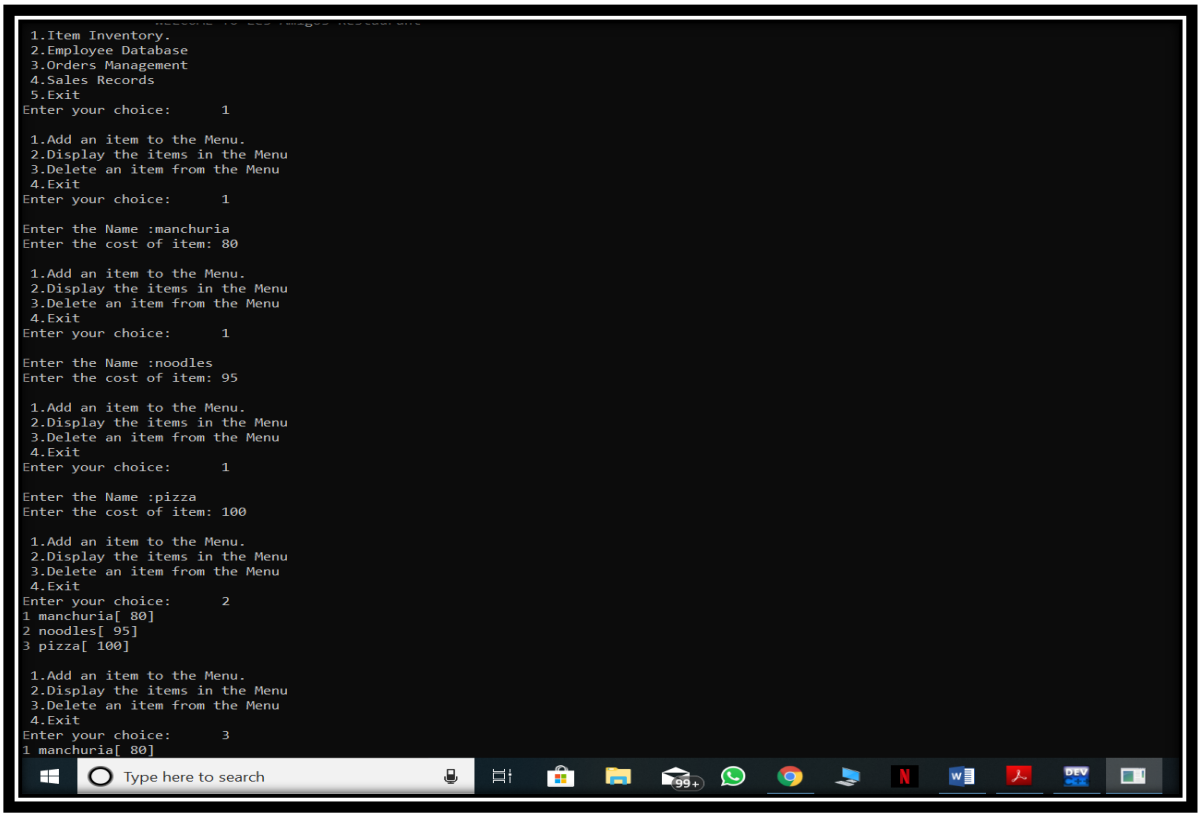
```
   for (temperory; temperory <= rear; temperory++)
   {
      printf("%s =>
%d.%d.%d\n",queue[temperory].name,queue[temperory].date,queue[temperory].month,q
ueue[temperory].year);
   }
}
void dequeue_order()
{
        if((rear==-1 && front==-1) || (rear<=front))
        {
                printf("\nNo 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;
          }
      }
}
```

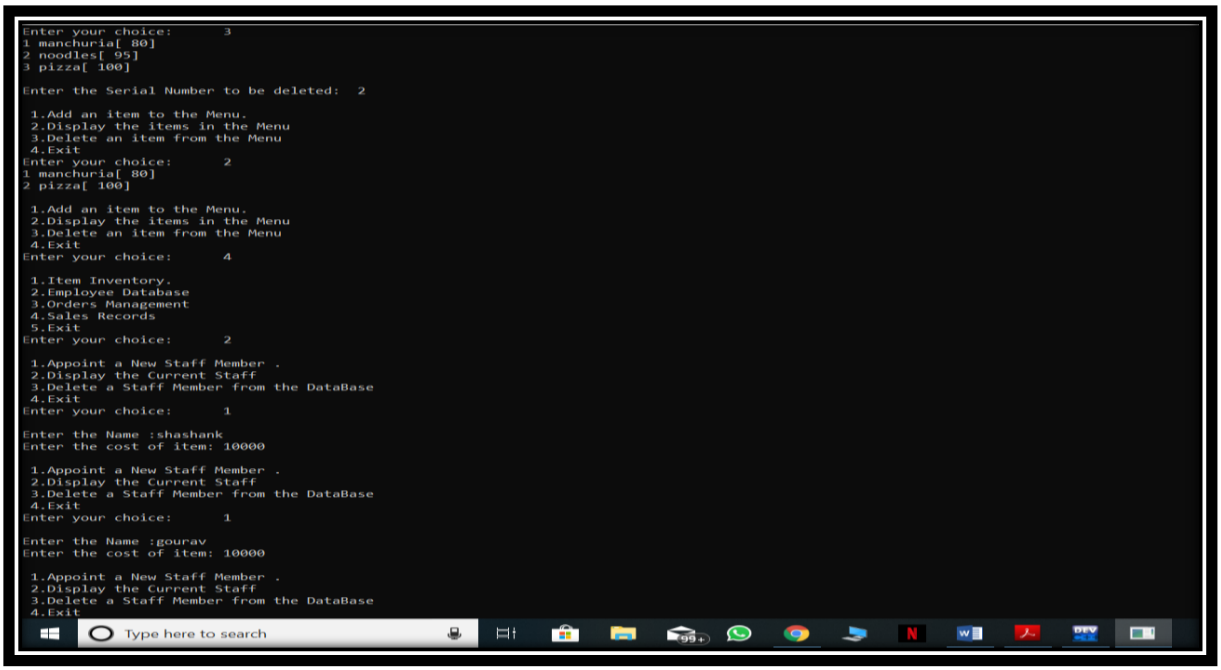
```
void item_menu()
{
        int choice=0;
    while(choice!=4)
                {
        printf("\n 1.Add an item to the Menu.\n");
        printf(" 2.Display the items in the Menu\n");
        printf(" 3.Delete an item from the Menu\n");
        printf(" 4.Exit\n");
        printf("Enter your choice:\t");
        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 Les-Amigos Restaurant----------------");
    int choice=0;
    while(1)
                {
        printf("\n 1.Item Inventory.\n");
        printf(" 2.Employee Database\n");
        printf(" 3.Orders Management\n");
        printf(" 4.Sales Records\n");
        printf(" 5.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:
                        sales_record();
                        break;
                case 5:
                        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 );
                  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);
}
```

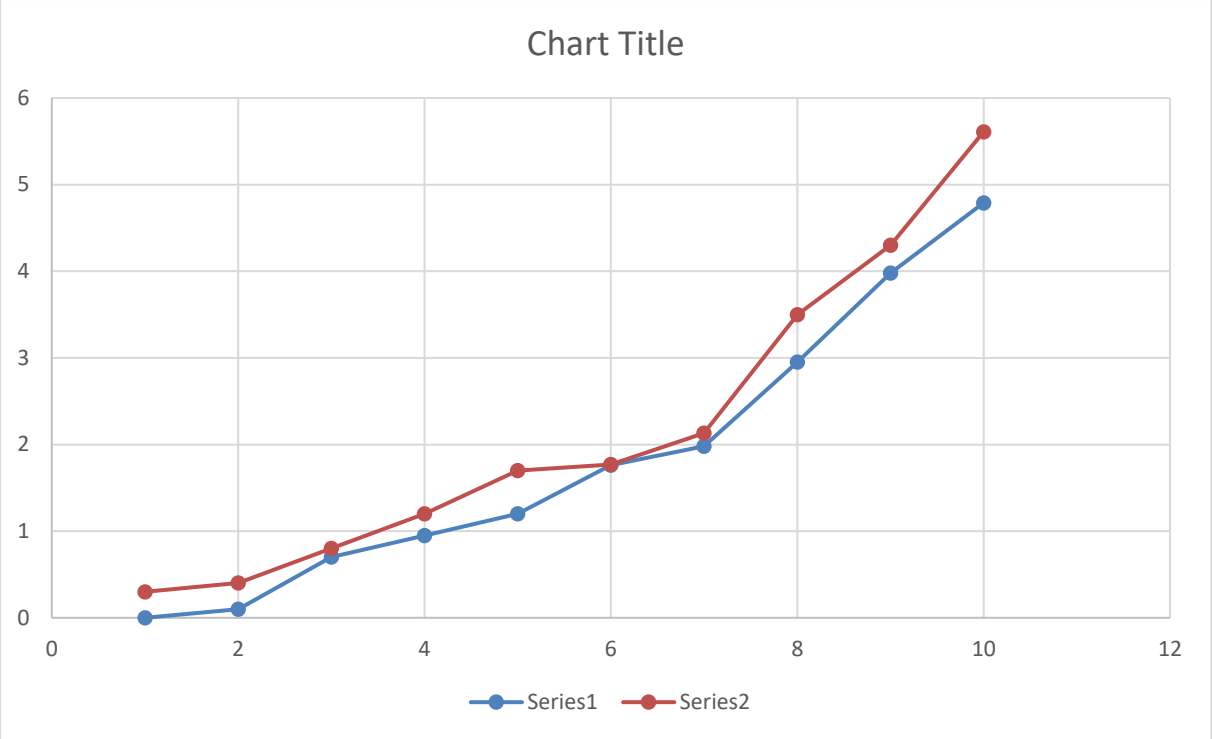




Graph

TABULATION:

INPUTS	TIME COMPLEXITY1	TIME COMPLEXITY2
1	0	0.3
2	0.1	0.4
3	0.7	0.8
4	0.95	1.2
5	1.2	1.7
6	1.76	1.77
7	1.98	2.13
8	2.95	3.56
9	3.86	4.3
10	4.79	5.61



Limitations

Any project isn't good. each project has some limitations. Our project conjointly has some limitations. These limitations could also be as a result of it's a mini project and that we are not associate degree professional. a number of the most limitations of the project or program area unit listed as follows:

¬ The show attraction of the program is poor. No attention-grabbing graphics area unit used as there's no use of within the program.

¬ No use of date and time operate.

¬ The choice of editable countersign isn't used considering the length of the program.

¬ The program solely focuses on providing choices to a manager and cashier however it cannot give a lot of of the choices for the client in an exceedingly supermarket.

so these area unit the most limitations of the comes. If these limitations will be overcome than our program would are an ideal one.

CONCLUSION

In our Project entitled “RESTAURANT asking MANAGEMENT SYSTEM” we've got tried our greatest to meet all the necessities of building.
The project being straightforward and versatile is running with success.
The main advantage of our project is that its simplicity attracts tons of users.
It is simply go by a novice user.
Our software can be used in any kind of restaurant (Sandwich Shop, Pizzeria, Steak House, Deli, Buffet, and Catering business, Doughnut or Pastry Shop, Hotel restaurant/kitchen and more).

The building asking Management System helps the building manager to manage the restaurant a lot of effectively and expeditiously by computerizing meal ordering, billing and
inventory control.
The system processes dealings and stores the ensuing knowledge.
Reports are generated from these knowledge that facilitate the manager to create applicable
business decisions for the restaurant. For example, knowing the number of customers
for a selected amount, the manager can decide whether more waiters and chefs are required.
This project once enforced it'll take away all the protection problems.
Also, there will be
speedy and secured authentication procedure for the maintenance of records. Data entry is fast and simple.
Therefore, our computer code will certainly convince be a made stepping stone 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. The objectives and scope of the project in future have been elaborated.

FUTURE SCOPE

Although the building business is extremely competitive, the life-style changes created by trendy living still fuel its steady growth.

More and additional individuals have less time, resources, and skill to cook for themselves.

Trends ar important and our package is well positioned for the present interests.

So whereas we have a tendency to confine mind the very fact that there's no package that's excellent, which might additionally mean there's no excellent building package, we have a tendency to place in AN
endless effort to improve our restaurant management software.

There area unit luxuriant options that don't seem to be enclosed during this project, thus we tend to area unit mentioning those options that may be additional in our project in future.

Those features are listed below-

• The Software can be used with a Web Based Interface.

• We offer flexible tax (VAT) for every item separately; our restaurant software can also supports multiple sales taxes in future which have the need to tender more than just one tax with their restaurant software.

Speaking of taxes, this building system will accommodates multiple excise systems (i.e.

USA, VAT, PST, ST, HST and much more).

• In future, when speaking of international features, our restaurant software can accommodates all world currencies. And if by any chance, your currency is not readily available in this restaurant software, you can simply add it yourself.

• In future, our restaurant software can be of "multilanguage" and "self translating" as well.

REFERENCE BOOKS

1) "ANSI C" by Balaguruswami.
2) Turbo C by Moolish Cooper.
3) Let Us C by Yashwant Kanetkar.
4) Working with C by Yaswant Kanetkar.
5) Programming in C by Hemant Goyal.