BANGALORE INSTITUTE OF TECHNOLOGY

K.R. Road, V.V. Puram, Bengalure – 560 004.

Branch : Computer Science Engineering.

Semester : ‘3rd’ Semester.

Section : ‘A’ .

Subject : Data Structures and Applications.

Subject code : 18CS32.

Mini Project Report

Submitted By:

Bharath Gowda.B	: 1BI19CS038
Bhuvan M	: 1BI19CS041
Harish.V	: 1BI19CS058
Suyog S Shimpukade	: 1BI19CS161

INDEX

INTRODUCTION

In this program we use linked list and queue data structures for implementation. Linked list a very commonly used linear data structure which consists of group of nodes in sequence. Each node holds its own data and the address of the next node hence forming chain like structure.

Linked lists were developed in 1955-1956 by Allen Newell , Cliff Shaw and Herbert A. Simon at RAND Corporation as the primary data structure for their Information Processing Language.

These nodes hold both the data and a reference to the next node in the list. Linked lists are often used because of their efficient insertion and deletion. They can be used to implement stacks, queues, and other abstract data types.

And we also use queue data structure in program. A queue is collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed is called the head or front of the queue. The operation of adding an element to the rear of the queue is known as Enqueue, and the operation of removing an element from the front is known as Dequeue.

PROBLEM STATEMENT

----

12. The car names are saved in a linked list. The customer orders are saved in the queue. You should take order from the beginning of the queue and search for it in the linked list, then delete it and put the sold car in the stack to be able to retrieve the last sold car.

MODULES

1. Single Linked List to Store Car details :

Using linked list concepts, new car details can be inserted to existing database. Node of the linked list is defined as below.

```
struct List
{
   char Name[MAX];
   int quantity;
   struct List *link;
};
typedef struct List *Node;
```

where ‘Name’ variable is used to store Car name.

‘quantity’ define quantity of car having same name.

- Add_Car( ) :

Add_Car( ) is use to add new car details to existing car list. Its takes car name as parameter and evaluate to produce output. If the parameter name matches the name of any of the existing cars then the quantity of that car will be increased by one or else the new car node is linked at rear end of the list. 'start' is the first node of the list.

```
Node start;

void Add_Car(char name[ ])
{
    Node temp,cur,pre;
    int d = 0;
    cur = start;
    MALLOC(temp,sizeof(struct List),Node);
    strcpy(temp->Name,name);
    temp->quantity = 1;
    temp->link = NULL;
    while(cur != NULL && d == 0)
    {
        if(strcmp(name,cur->Name) == 0 )
        {
            cur->quantity++;
            d = 1;
        }
        pre = cur;
        cur = cur->link;
    }
    if(d == 0)
    {
        CarCount++;
        if(start == NULL)
            start = temp;
        else
            pre->link = temp;
    }
}
```

- Display_Car_List( ) :

This function is used to display the list of cars available.

```
void Display_Car_List( )
    {
        Node cur;
        int i;
        clrscr();
        printf("\t\t\t\t%s\t\t\n",ShowRoomName);
        printf("-------------------------------------------------------------------------------\n");
        cur = start;
        for( i = 1 ; cur != NULL; i++ )
        {
                printf("%d. %s \t\t : %d\n",i,cur->Name,cur->quantity);
                cur = cur->link;
        }

    }
```

2. Stack to Retrieve Last Sold Car Detail :

Array of strings concept is used to implement stack. The stack contains the car name that are previously sold. The stack can store upto last 30 sold car values. But the given problem we are only taking the last one which is at the top of the stack.

```
char  s[MAX]{MAX] ;
int top;    //which points the top element of the stack ‘s’
```

The last sold car can be retrieved , which is stored at the top of the Stack.

3. Circular Queue to Manage Customer Order :

Circular queue is created using array of string concepts, which is used to manage customer order, where orders are processed from the front end and new orders are inserted at rear end. Since it is a circular queue the 'MAX' defines the max. no of orders that can be stored, if the no. of pending orders exceed this value , the oldest order will be replaced by the newest one.

```
char q[MAX][MAX];
int front , rear, OrderCount;
```

where 'OrderCount' variable keeps the count of pending orders. 'front' points the front end of the queue 'q' , and 'rear' points the rear end of 'q'.

- void Dequeue()
```
  {
    front = (front+1)%MAX;
    OrderCount--;
  }
```
- void Enqueue(char name[])
```
  {
       rear = (rear+1)%MAX;
         strcpy(q[rear],name);
         OrderCount++;
         printf("Ordered placed successfully\nThank You\n");
         getch();
  }
```

Dequeue( ) is used to delete front item of the queue.
Enqueue( ) is used to insert new order to the rear end.

- Pending_Orders( ) :

  Another function Pending_Orders( ) is defined based on the logics to give expected output. It is defined as such that the order can be either accepted or rejected. If the order is accepted but out of stock, the order will be deleted at front and inserted at the rear, so that other orders can be processed first.

  void Pending_Orders( )

```
{
    char CurrentOrder[30];
    clrscr();
    printf("\t\t\t\t%s\t\t\n",ShowRoomName);
    printf("------------------------------------------------------------------------\n");
    printf("\nNo. of orders in Pending : %d\n",OrderCount);
    if(OrderCount == 0)
    {
        printf("\nNo Pending Orders ('-')\n");
        getch();
        return;
    }
    strcpy(CurrentOrder,q[front]);
    printf("CurrentOrder : %s\n",CurrentOrder);
    printf("1. Accept Order\n");
    printf("2. Reject Order\n\n");
    printf("3. Back\n\n");
    printf("Enter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1 : Delete_List(CurrentOrder);
                    break;
          case 2 : printf("Order Rejected\n");
                   front = (front + 1)%MAX;
                   OrderCount--;
                   getch();
                   return;
         case 3 : return;
         default: printf("Invalid Choice\n");  }
}
```

- Delete_List ( ) : This function is used to delete car form the list, where it takes the order (i.e car name) and traverse through the list. If car exists in the list and its quantity is more than 1, then the quantity is reduced by one, or else the car node will be deleted from the list.

```
void Delete_List(char order[])
 {   Node cur,pre;
     int flag=0;
     cur = start;
     pre = cur;
     while(cur != NULL)
     {     if(strcmp(cur->Name,order) == 0)
           {  if(cur->quantity <= 1)
                {    pre->link = cur->link;
                     if(start == cur)
                          start = cur->link;
                     free(cur); }
              else
                     cur->quantity--;
              flag = 1;
              Dequeue();
              break;    }
              pre = cur;
              cur = cur->link;   }
     if(flag == 0)
          {      printf("Out of Stock.\nRearranging the Pending orders :\n");
                 rear = (rear + 1)%MAX;
                 strcpy(q[rear],q[front]);
                 front = (front + 1)%MAX;
                 getch();   }
     else
          {    top = (top+1)%MAX;
               strcpy(s[top],order);
               printf("Order Accepted\n");
               CarCount--;
               getch( );    }}
```

4. User Interface Modules :

The program works on menu driven events. So having different modules for different menu will make the program easier and more understandable.

- Main_UI( ) :

This one is the main menu or the starting menu , where there are three option. One for administrative access, one for customer use and last one to exit the program.

```
void Main_UI()
{
    clrscr();
    printf("\t\t\t\t%s\t\t\n",ShowRoomName);
    printf("------------------------------------------------------------------------\n");
    printf("1. Adminstrative Access\n");
    printf("2. Customer Access\n\n");
    printf("3. Exit\n\n");
    printf("Enter your Choice : ");
    scanf("%d",&choice);
    switch(choice)
      {
        case 1 : Admin_UI();
                   break;
        case 2 : Customer_UI();
                   break;
        case 3 : done=1;;
                   break;

        default: printf("Invalid Choice");
      }
 }
```

- Admin_UI( );

This one occurs when the user choice is 1 in the Main_UI( ). We have five options , each option have their own functions which are already explained.

```
void Admin_UI()
{
        int y = 0;
        char name[MAX];
        while(y==0)
         {
                clrscr();
                printf("\t\t\t\t%s\t\t\n",ShowRoomName);
                printf("-----------------------------------------------------------------\n”);
                printf("1. Available Car List\n");
                printf("2. Pending Orders\n");
                printf("3. Last Sold Car\n");
                printf("4. Add New Car\n\n");
                printf("5. Back \n\n");
                printf("Enter your Choice : ");
                scanf("%d",&choice);

                switch(choice)
                {
                    case 1 : Display_Car_List();
                             getch();
                             break;
                    case 2 : Pending_Orders();
                             break;
                    case 3 : printf("The last sold car is : %s ",s[top]);
                            getch();
                            break;
                    case 4 : printf("Name : ");
                            scanf("%s",name);
                            Add_Car(name);
                            break;
                    case 5 : y = 1;
                             break;
                    default: printf("Invalid Choice");

                }
         }
}
```

- Customer_UI( ) :

This menu occurs if the user choice is 2 in the Main_UI( ). This Menu is for customer use where they can view available cars and even place order.

```
void Customer_UI()
{
   int x =0,c_choice,i;
   char c;
   Node cur;
   while(x==0)
   {
         clrscr();
         cur = start;
         printf("\t\t\t\t%s\t\t\n",ShowRoomName);
         printf("--------------------------------------------------------------------\n");
         printf("1. Available Car List\n\n");
         printf("2. Back\n\n");
         printf("Enter your Choice :");
         scanf("%d",&choice);
         switch(choice)
         {
            case 1 : Display_Car_List();
                        printf("Book Car w.r.t  Number : ");
                        scanf("%d", &c_choice);
                        if(c_choice>0 && c_choice <= CarCount)
                        {
                                for(i = 1 ; i< c_choice;i++)
                                {
                                  cur = cur->link;
                                }
                                printf("Confirm your order for %s (y/n) : ",
                                cur->Name);
                                scanf(" %c",&c);
                                if(c == 'y')
```

```
                                          {
                                           Enqueue(cur->Name);
                                          }
                                          else
                                          {
                                             printf("Order not placed\n ");
                                             getch();
                                          }
                                 }
                                 else
                                 {
                                    printf("Invalid Choice, Getting Back\n");
                                    getch();
                                 }
                                 break;
                    case 2 :   x = 1;
                               break;
                    default: printf("Invalid Choice");

                 }
            }
      }
```

5.Initialization( ) :

The main purpose of this function is to initialize some values to car list and order queue. So that they have some initial values when the program is executed and can be used to perform different operations.

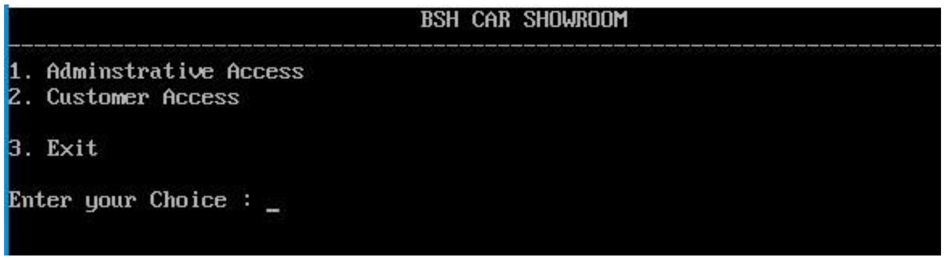
```
void Initialization()
{
  char Cars[5][30]={"Fortuner","Vitara Breeza","Swift","Swado","Innova"  };
  char Order[2][30]={"Vitara Breeza","Swift"};
  int i ;
  strcpy(s[top],"Nil");
  for(i =0 ;i<5;i++)
     Add_Car(Cars[i]);
  for(i=0;i<2;i++)
  {
     rear = (rear+1)%MAX;
     strcpy(q[rear],Order[i]);
     OrderCount++;
  }

}
```

IMPLEMENTATION

The program is developed on basics of **Test driven development.** Each module is developed separately and test, later linked to form the whole program.

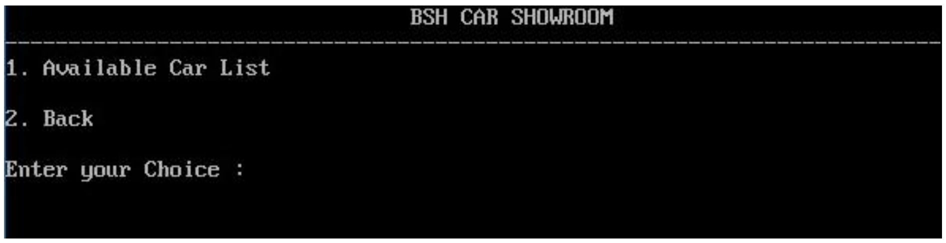
OUTPUT SNAPSHOT

- Main Menu :



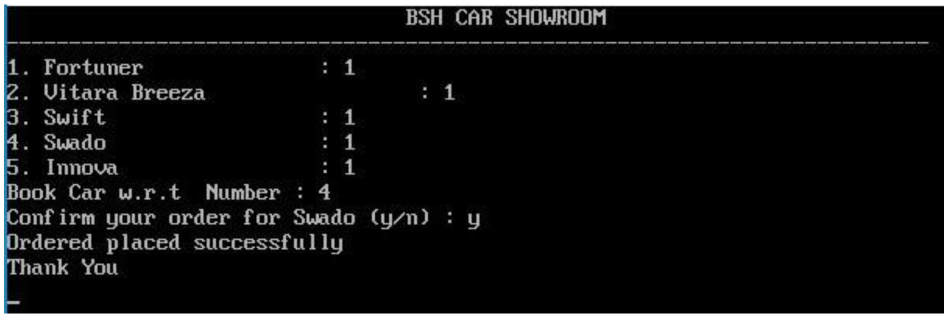
```
                                   BSH CAR SHOWROOM
------------------------------------------------------------------------------------
1. Adminstrative Access
2. Customer Access

3. Exit

Enter your Choice : _
```

- Customer Menu :



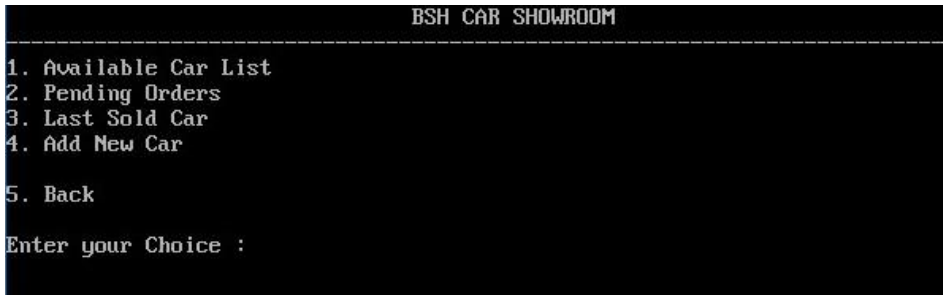
```
                                  BSH CAR SHOWROOM
------------------------------------------------------------------------------------
1. Available Car List

2. Back

Enter your Choice :
```

- Placing Order :



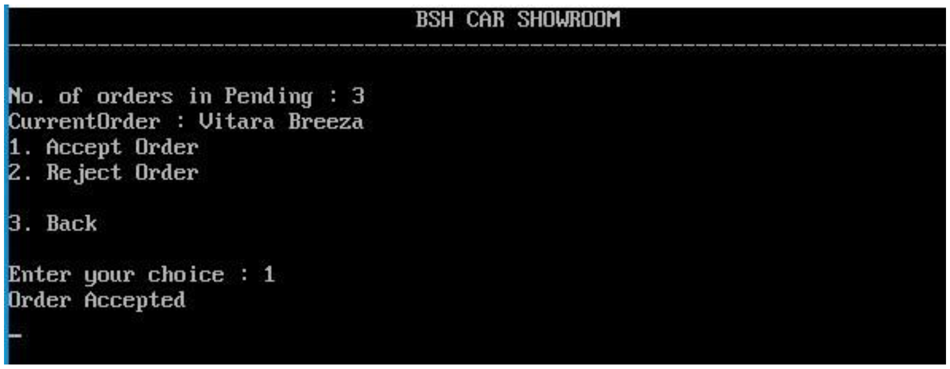
```
                                  BSH CAR SHOWROOM
----------------------------------------------------------------------------------
1. Fortuner               : 1
2. Vitara Breeza                  : 1
3. Swift                  : 1
4. Swado                  : 1
5. Innova                 : 1
Book Car w.r.t  Number : 4
Confirm your order for Swado (y/n) : y
Ordered placed successfully
Thank You
_
```

- Administrator Menu :



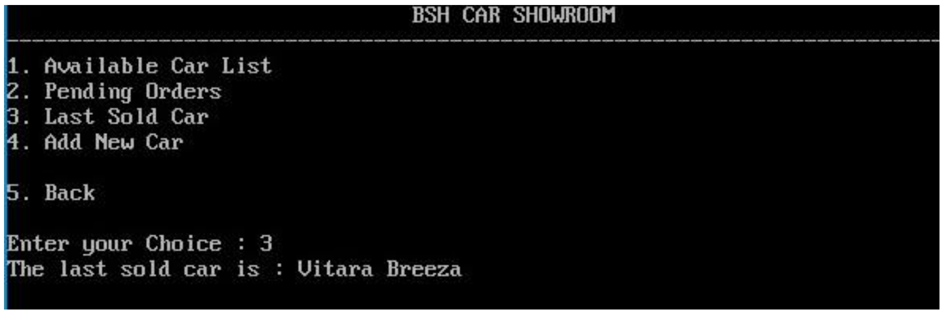
```
                                  BSH CAR SHOWROOM
------------------------------------------------------------------------------------
1. Available Car List
2. Pending Orders
3. Last Sold Car
4. Add New Car

5. Back

Enter your Choice :
```

- Pending Orders :



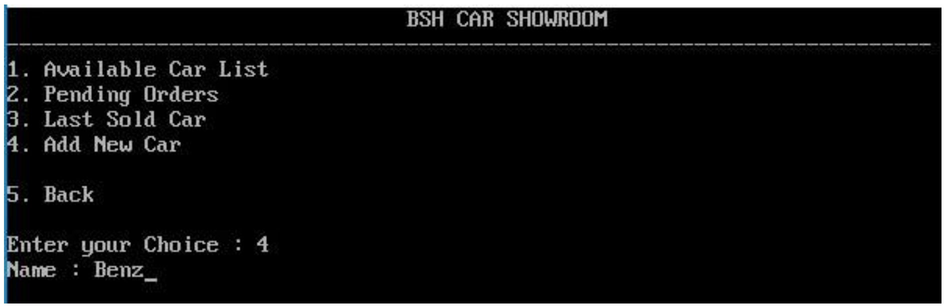
```
                              BSH CAR SHOWROOM
-----------------------------------------------------------------------------

No. of orders in Pending : 3
CurrentOrder : Vitara Breeza
1. Accept Order
2. Reject Order

3. Back

Enter your choice : 1
Order Accepted
_
```

- Last Sold Car :



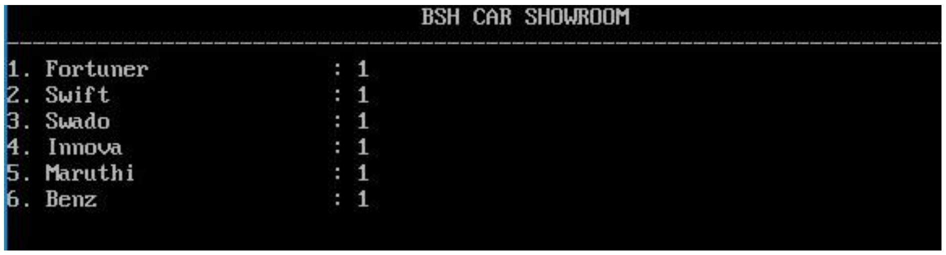
```
                              BSH CAR SHOWROOM
-----------------------------------------------------------------------------
1. Available Car List
2. Pending Orders
3. Last Sold Car
4. Add New Car

5. Back

Enter your Choice : 3
The last sold car is : Vitara Breeza
```

- Add new Car :



```
                              BSH CAR SHOWROOM
-----------------------------------------------------------------------------
1. Available Car List
2. Pending Orders
3. Last Sold Car
4. Add New Car

5. Back

Enter your Choice : 4
Name : Benz_
```

- Availabel Car List (Admin Menu) :



```
                              BSH CAR SHOWROOM
-----------------------------------------------------------------------------
1. Fortuner              : 1
2. Swift                 : 1
3. Swado                 : 1
4. Innova                : 1
5. Maruthi               : 1
6. Benz                  : 1
```

APPLICATION

- Can be used by small scale car sellers.
- Can also be used by large scale car sellers , with some small changes and a separated database, so that all branches can be linked together .
- The program is not limited to just cars, it can be used for any type of Automobile.