# **19CSE102 – C PROGRAMMING**

# **PROJECT EVALUATION**

# **TOPIC : AUTOMOBILE ASSEMBLY PLANT**

**STUDENT DETAILS**

**PREETHI PRABHA – CB.EN.U4CSE20051**

**P. KOMAL HARSHITH – CB.EN.U4CSE20047**

**SAKETH GOUD – CB.EN.U4CSE20066**

**B. VISHNU DHEERAJ – CB.EN.U4CSE20013**

**PROBLEM STATEMENT :**

To build a C program to build a menu-driven application for an automobile assembly unit, where the customers are vehicle dealers . The program should have provisions for multiple dealers to make purchases, and should provide a menu of tasks for the manager to choose from.

**FUNCTIONALITIES :**

* P. KOMAL HARSHITH – Structure and Variable Declarations
  + Use of Dynamic Memory Allocation and Pointers to allocate required memory based on user input
  + Three structures for easy handling
* SAKETH GOUD – Manager Options
  + Do while loop to keep repeating the module as needed
  + Print available stock in the unit
  + Close the application
* B. VISHNU DHEERAJ – Dealer Options
  + Print available stock in the unit
  + Update available quantity upon purchase
  + Take dealer input for purchase details
* PREETHI PRABHA – Functions
  + Function to accept an amount in the register and update the register
  + Function to find dealer purchase details with the dealer ID
  + Function to print maximum and minimum purchase made by the dealers

**C CODE :**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <math.h>

#include <string.h>

float acceptAmount(struct dealer\* x, float y);

int MaxPurchase(struct dealer\* x, int n);

int FindDealer(struct dealer\* x, int n, int y);

struct dealer

{

int dealerID;

char carname[20];

int carqty;

float purchaseamount;

};

struct manager

{

char password[10];

int close;

int repeat;

};

struct cardetails

{

char carname[20];

int carqty;

float carprice;

};

int main()

{

float amount = 0;

int user , purchase , man ;

char model[100];

char pwd[] = "HELLO";

char designation[9];

int dID;

int ManOptions;

struct manager m;

/\*Declarations - Komal Harshit\*/

struct dealer\* deal;

struct cardetails \* car;

int i, n, z, j = 0;

printf("Enter the types of car models: ");

scanf("%d",&n);

// allocating memory for n numbers of struct customer;

car = (struct cardetails\*)malloc(n \* sizeof(struct cardetails));

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

{

printf("Enter car name , available quantity of stock and price per car : ");

/\*To access members of 1st struct dealer,

deal->carname , deal->carqty deal->purchaseamount is used\*/

scanf("%s %d %f",(car + i)->carname,&(car + i)->carqty,&((car + i)->carprice));

}

printf("Enter the permitted number of dealers : ");

scanf("%d",&z);

// allocating memory for z numbers of struct customer;

deal = (struct dealer\*)malloc(z \* sizeof(struct dealer));

do /\*Main Do while loop\*/

{

/\*Manager Options - Saketh goud\*/

printf("\nEnter designation (dealer / manager) : ");

scanf("%s", designation);

if ((strcmp(designation, "manager")) == 0)

{

printf("Enter the password: ");

scanf("%s", m.password);

do // Do while if password is incorrect and continue is executed

{

if (strcmp(m.password, pwd) != 0)

{

printf("Enter correct password");

continue;

}

else

printf("\nIs the plant ready to close for the day (1 for yes / 0 for no) : ");

scanf("%d", &m.close);

do //Do while for manager options

{

if (m.close == 1)

{

printf("\nChoose one of the following options: \n1) .Output account balance\n2) View purchase details for dealers\n3) .Output highest and lowest purchase details\n4) Output available quantity of each item\n5) Close the application\n");

scanf("%d", &ManOptions);

switch (ManOptions)

{

case 1: printf("Amount in register : %.2f", amount);

break;

case 2: do

{

printf("\nEnter the dealer ID : ");

scanf("%d", &dID);

FindDealer(deal, z, dID);

printf("\nDo you want to continue? (1 for yes / 0 for no)");

scanf("%d", &m.repeat);

} while (m.repeat != 0);

break;

case 3: MaxPurchase(deal, z);

break;

case 4: printf("\nCar Name Available Quantity Price per car ");

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

{

printf("\n%s\t\t %d\t\t%.2f", (car + i)->carname, (car + i)->carqty, (car + i)->carprice);

}

break;

case 5: printf("The plant is closed for the day");

exit (0) ;

break;

default: break;

}

}

printf("\nDo you want to go back to the user choice screen ? (1 for yes / 0 for no)");

scanf("%d", &man);

} while (man == 1);

} while (strcmp(m.password, pwd) != 0);

}

else if ((strcmp(designation, "dealer")) == 0)

{

do /\*Do while for dealer purchase\*/

{

printf("\nCar Name Available Quantity Price per car ");

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

{

printf("\n%s\t\t %d\t\t%.2f",(car + i)->carname, (car + i)->carqty, (car + i)->carprice);

}

printf("\nEnter chosen car : ");

scanf("%s", (deal + j)->carname);

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

{

if ((strcmp((car + i)->carname, (deal + j)->carname)) == 0)

{

printf("\nAvailable Stock : %d \nPrice : %.2f", (car + i)->carqty, (car + i)->carprice);

printf("\nEnter Dealer ID :");

scanf("%d", &(deal + j)->dealerID);

printf("\nEnter purchase quantity : ");

scanf("%d", &(deal + j)->carqty);

if ((deal + j)->carqty <= (car + i)->carqty)

{

(deal + j)->purchaseamount = (deal + j)->carqty \* (car + i)->carprice;

printf("\nTotal Cost : Rs.%.2f", (deal + j)->purchaseamount);

amount = acceptAmount(deal + j, amount);

(car + i)->carqty = ((car + i)->carqty) - ((deal + j)->carqty);

printf("\nOrder will be dispatched soon");

j++;

}

else

printf("\nEnter correct quantity");

}

}

printf("\nDo you want to make another purchase ? (1 for yes / 0 for no) ");

scanf("%d", &purchase);

} while (purchase == 1 && (j<z));

}

else

printf("\nEnter correct designation");

printf("\nDo you want to go back to the main screen ? (1 for yes / 0 for no)");

scanf("%d", &user);

} while (user == 1);

return 0;

}

/\*Funtions - Preethi Prabha\*/

float acceptAmount(struct dealer\* x, float y)

{

float amt;

do {

printf("\nEnter the amount to confirm payment : ");

scanf("%f", &amt);

if ((x->purchaseamount) == amt)

{

amt = x->purchaseamount;

y = y + amt;

return y;

}

else

printf("\nEnter correct amount");

} while ((x->purchaseamount) != amt);

}

int MaxPurchase(struct dealer\* x, int n)

{

int maxindex = 0, minindex = 0;

float maxpur = x->purchaseamount;

float minpur = x->purchaseamount;

for (int i = 1; i < n ; i++)

{

if ((x + i)->purchaseamount > maxpur)

{

maxpur = (x + i)->purchaseamount;

maxindex = i;

}

else if ((x + i)->purchaseamount < minpur)

{

minpur = (x + i)->purchaseamount;

minindex = i;

}

}

printf("\nMaximum purchase is by dealer ID %d", (x + maxindex) ->dealerID);

printf("\nMinimum purchase is by dealer ID %d", (x + minindex)-> dealerID );

return 0;

}

int FindDealer(struct dealer\* x, int n, int d )

{

int i;

for (int i = 0; i < n; i++)

{

if ((x + i)->dealerID == d)

printf("\n%s %d %.2f",((x + i)->carname), (x + i)->carqty, (x + i)->purchaseamount);

}

return 0;

}

**SAMPLE OUTPUT:**









