Ans_Sheet_4

- في الشيت ده هنعمل program نطبق فيه علي ال stack and queue مع بعض

- Write a C program that keeps track of the customers visiting a car workshop. The program utilizes two data structures, a stack and a queue to have customers' data in particular orders.

  The main program should display the following menu :

  1. Add a New Customer.
  2. Serve a Customer.
  3. Display Customers Information.
  4. Display Customers information in the "most-recent"Order.
  5. Exit menu

* By choosing "Add a New Customer" you should enter the data of the new arriving customer and save it such that he has the least priority among others.

* By choosing "Serve a Customer" you should display the data of the first arriving customer then dismiss them from the system.

* "Display Customers Information" prints on screen the data of the current waiting customers without serving them.

* "Display Customers in the most-recent Order" without serving them - should be done by copying the data to a structure that reverses the order.

Ans

Stack .c

```c
#include "stack.h"

void CreateStack(Stacktype *s)
{
    s->top = 0;
}

void Push(Stacktype *s , Customer item)
{
    s->arr[s->top++] = item;
}

void Pop(Stacktype *s , Customer *item)
{
    *item = s->arr[--s->top];
}

int Stackempty(Stacktype s)
{
    return(s.top == 0);
}

int Stackfull(Stacktype s)
{
    return(s.top == Size);
}
```

Stack .h

```c
#ifndef STACK_H_INCLUDED
#define STACK_H_INCLUDED
#include "global.h"

typedef struct
{
    int top;
    Customer arr[Size];

}Stacktype;

void CreateStack(Stacktype *s);
void Push(Stacktype *s , Customer item);
void Pop(Stacktype *s , Customer *item);
int Stackfull(Stacktype s);
int Stackempty(Stacktype s);

#endif // STACK_H_INCLUDED
```

Queue .c

```c
#include "queue.h"

void CreateQueue(QueueType *q)
{
    q->front = 0;
    q->rear = Size - 1;
    q->size = 0;
}

int IsEmpty(QueueType *q)
{
    return (q->size == 0);
}

int IsFull(QueueType *q)
{
    return(q->size == Size);
}


void Enqueue(QueueType *q , Customer c)
{

    q->rear = (q->rear + 1)% Size;
    q->Q[q->rear] = c;
    q->size ++;

}

void Dequeue(QueueType *q , Customer *c)
{
    *c = q->Q[q->front];
    q->front = (q->front + 1) % Size ;
    q->size --;
}
```

Queue .h

```c
#ifndef QUEUE_H_INCLUDED
#define QUEUE_H_INCLUDED
#include "global.h"

typedef struct
{
    int front;
    int rear;
    int size;
    Customer Q[Size];
}QueueType;

void CreateQueue (QueueType *q);
void Enqueue (QueueType *q , Customer c);
void Dequeue (QueueType *q , Customer *c);
int IsFull(QueueType *q);
int IsEmpty (QueueType *q);


#endif // QUEUE_H_INCLUDED
```

global .h

```h
#ifndef GLOBAL_H_INCLUDED
#define GLOBAL_H_INCLUDED
#define Size 5

typedef struct
{
    int id;
    char name[5];
}Customer;


#endif // GLOBAL_H_INCLUDED
```

Main .c

```c
#include <stdio.h>
#include <stdlib.h>
#include "queue.h"
#include "stack.h"

int main()
{
    int Choice,i,j;
    Customer cst[Size] , n , display,dis[Size];
    QueueType q;
    Stacktype s;
    CreateQueue(&q);
    CreateStack(&s);

    for(i = 0 ; i < Size - 1 ; i++)
    {
        scanf("%d",&cst[i].id);
        scanf("%s",cst[i].name);
        Enqueue(&q,cst[i]);
        Push(&s,cst[i]);
    }


    printf("\n Enter 1. To Add A New Customer") ;
    printf("\n Enter 2. To Serve A Customer ") ;
    printf("\n Enter 3. To Display Customers Information") ;
    printf("\n Enter 4. Display Customers information in the most-recent Order ") ;
    printf("\n Enter 5. To Exit Menu ") ;
    printf("Enter Your Choice : ") ;
    scanf("%d",&Choice);

```

```
31
32      switch(Choice)
33      {
34          case 1:
35          {
36              printf("Enter New Customer ID : ");
37              scanf("%d",&n.id);
38              printf("Enter New Customer Name : ");
39              scanf("%s ",n.name);
40              Enqueue(&q,n);
41              break;
42          }
43
44          case 2:
45          {
46              Dequeue(&q,&display);
47              printf("\nCustomer ID : %d\n",display.id);
48              printf("Customer Name Is : %s\n",display.name);
49              break;
50          }
51
52          case 3:
53          {
54              Dequeue(&q,&display);
55              printf("\n%d\n",display.id);
56              printf("%s",display.name);
57              Enqueue(&q,display);
58              break;
59          }
60
61          case 4:
62          {
63              for(i = 0 ; i < Size - 1 ; i ++)
64              {
65                  Pop(&s,&dis[i]);
66                  printf("%d\t",dis[i].id);
67                  printf("%s\n",dis[i].name);
68
69              }
70              break;
71
72          }
73
74          case 5:
75          {
76              return 0;
77          }
78      }
79
80      return 0;
81  }
```