Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology

Name :Dasari Venkata Sai Rithvik .
Email :rithvikdasari1011@gmail.com
Roll no:23071A05N9
Phone :7670963113
Branch :VNRVJIET
Department :CSE
Batch :2027
Degree :B.Tech CSE

2023_27_II_CSE 4_Data Structures_lab

DATA STRUCTURES_ASSESSMENT_WEEK 3

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : CODING

1. Problem Statement

A company is designing an automated parking system for a multi-level parking garage. The system should be able to park cars in an available spot and also retrieve cars from the garage. The parking garage has a limited number of spots, so the system should be able to keep track of which spots are available and which ones are occupied. The company wants to implement this using a circular queue.

We can implement this using a circular queue data structure where each node in the queue represents a parking spot. Initially, all the spots will be empty, and as cars are parked, the nodes in the queue will be updated to indicate that the spot is occupied. When a car leaves, the node will be updated to indicate that the spot is empty.

Implement a menu-driven program. The menu options are:

Park a car

Retrieve a car.

Exit

default: invalid choice

If the parking lot is full and the user attempts to park, the message "Parking lot is full" should appear.

If the parking lot is empty and the user attempts to park, the message "Parking lot is empty" should appear.

Note: There is a new line space after displaying the last line of the output.

Answer

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 10
int front=-1,rear=-1,queue[SIZE];
void enqueue();
void delqueue();
int main()
{
   int choice;
   do{
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:enqueue();break;
         case 2:delqueue();break;
         case 3:break;
         default:printf("Invalid choice.\n");
      }
   }while(choice!=3);
   return 0;
}
void enqueue()
{
   int item;
   if(( front == rear+1 ) || front == 0 && rear == SIZE-1 )
   {
      printf("Parking lot is full.\n");
   }
```

```
    else
    {
        if( front == -1 )
        front=0;
        scanf("%d",&item);
        rear=(rear+1)%SIZE;
        queue[rear]=item;
        printf("Car %d parked in spot %d.\n",item,rear);
    }
}
void delqueue()
{
    int item,i;
    i=front;
    if(front==-1)
    {
        printf("Parking lot is empty.\n");
    }
    else
    {
        item=queue[front];
        if( front == rear )
        {
            front=rear=-1;
        }
        else
        {
            front=(front+1)%SIZE;
        }
        printf("Car %d retrieved from spot %d.\n",item,i++);
    }
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Design and implement a food ordering system for a restaurant using a circular queue data structure. The system should manage incoming food orders from multiple customers and allocate kitchen staff to prepare the

orders efficiently.

The system should allow the user to perform the following operations:

Enqueue a food order into the queue with a unique order number, customer name, and list of food items ordered.

Dequeue an order from the queue (when it has been processed by the kitchen staff).

Display the current state of the queue with the order numbers, customer names, and list of food items ordered.

Exit the program.

The program should be able to handle invalid inputs and display appropriate error messages. The program should also be able to handle the circular nature of the queue, ensuring that when the last element is reached, the next enqueued order is added at the beginning of the queue.

Note: There is a new line space after the last line of the output.

Answer

```
#include<stdio.h>
#include<stdlib.h>
#define size 100
struct order{
   int order_num;
   int num;
   char food_list[20][100];
   char name[100];
}q[100];
int f=-1,r=-1;
void insert();
void delet();
void show();
int main()
{
   int choice;
   do{
      scanf("%d",&choice);
```

```
        switch(choice)
        {
           case 1:insert();break;
           case 2:delet();break;
           case 3:show();break;
           case 4:break;
           default :printf("Invalid choice\n");
        }
    }while(choice!=4);
    return 0;
}
void insert()
{
  int i,j,n;
  if((f==r+1) || (f==0 && r==size-1))
   printf("Queue is full\n");
  else
  {
      if(f==-1)
       f=0;
      r=(r+1)%size;
    scanf("%d",&q[r].order_num);
    i=q[r].order_num;
    scanf("%s",q[r].name);
    scanf("%d",&q[r].num);
    j=q[r].num;
    for(n=0;n<j;n++)
    scanf("%s",(q[r].food_list[n]));
    printf("Order added to queue\n");
  }

}
void delet()
{
    int i;
    if(f==-1)
    printf("Queue is empty\n");
    else
    {
        printf("Order number: %d\nCustomer name: %s\n",q[f].order_num,q[f].name);
        printf("Food items:\n");
        for(i=0;i<q[f].num;i++)
```

```
        printf("%s\n",q[f].food_list[i]);
        if(f==r)
         f=r=-1;
        else
         f=(f+1)%size;
    }

}
void show()
{
    int i,j;
    if(f==-1)
    printf("Queue is empty\n");
  else
    {
      printf("Order number  Customer name  Food items\n");
      for(i=f;i!=r;i=(i+1)%size)
      {
       printf("%d   %s  ",q[f].order_num,q[f].name);
       for(j=0;j<q[f].num;j++)
        printf("%s,",q[f].food_list[j]);
        printf("\n");
       }
       printf("%d   %s ",q[r].order_num,q[r].name);
       for(j=0;j<q[r].num;j++)
       printf("%s,",q[r].food_list[j]);
       printf("\n");
    }
}
```

Status : Correct ***Marks : 10/10***