Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Information Technology

Department of

Artificial Intelligence and Data Science

Name: Siddhesh Dilip Khairnar		
Class: SY	Division: B	Roll No: 272028
Semester: III	Academic Year: 2022-2023	
Subject Name & Code: Data Structure, ADUA21202		
Title of Assignment: Perform implementation of Queue using array and Linked List Enqueue, Dequeue		

Assignment No.- 7

PAGE NO.:
DATE: / /

DS Assignment 7

Name: Siddhesh Khairnar
PRNno: 22110398 Roll no: 272028

Aim: Perform implementation of queue using array i) Enquence & ii) Dequence

Theory: To implement a queue using an array,
- create an array arr of size n and
- take two variable front and rear both of which will be initialized to 0 which means the queue is currently empty.
- Element.

• rear is the index up to which the element are stored in the array &
• front is the index of the first element of the array.

Now, some of the implementation of queue operation are as follows:

Enquene: Addition of an element to the queue. Adding an element will be performed after checking whether the queue is full or not. If rear < n which indicates that the array is not full then store the element at arr [rear] and increment rear by 1 but if rear == n the it is said to be an overflow condition as the array is full.

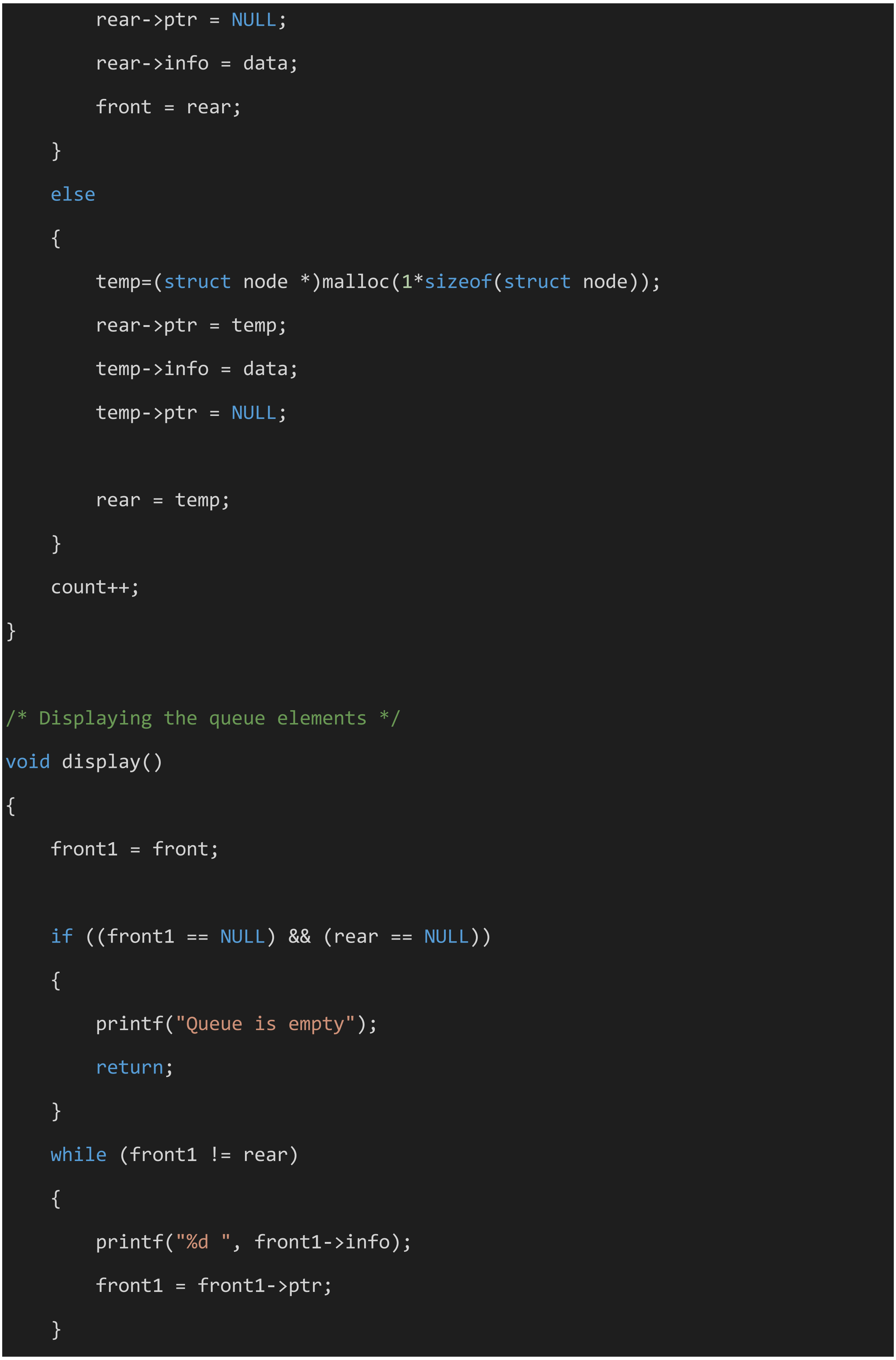
Dequeue: Removal of an element from the queue. An element can only be deleted when there is at least an element to delete that rear > 0. Now, the element at arr [front] can be deleted but all the remaining element have to shift to the left on another dequeue operation.

Conclusion: Thus we have successfully implemented queue using array
(i) Enqueue
(ii) Dequeue.

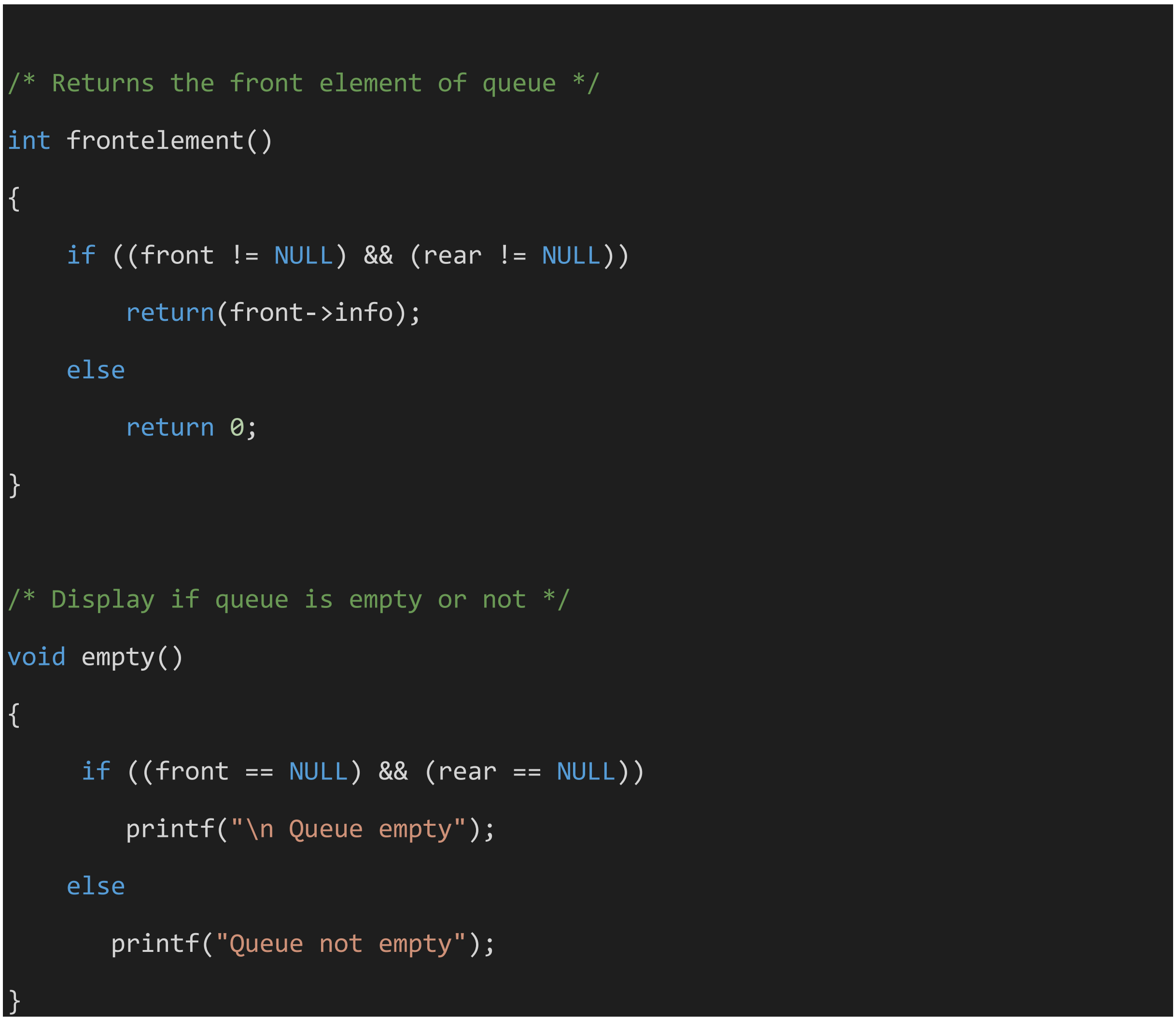
Ⓒ
Ashwin

Program:

```c
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int info;
    struct node *ptr;
}*front,*rear,*temp,*front1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

int main()
{
    int no, ch, e;

    printf("\n 1 - Enque");
    printf("\n 2 - Deque");
    printf("\n 3 - Front element");
    printf("\n 4 - Empty");
    printf("\n 5 - Exit");
    printf("\n 6 - Display");
    printf("\n 7 - Queue size");
    create();
    while (1)
    {
        printf("\n Enter choice : ");
        scanf("%d", &ch);
        switch (ch)
{
        case 1:
            printf("Enter data : ");
            scanf("%d", &no);
            enq(no);
            break;
        case 2:
            deq();
            break;
        case 3:
            e = frontelement();
            if (e != 0)
                printf("Front element : %d", e);
            else
                printf("\n No front element in Queue as queue is empty");
            break;
        case 4:
            empty();
            break;
        case 5:
            exit(0);
        case 6:
            display();
            break;
        case 7:
            queuesize();
            break;
        default:
            printf("Wrong choice, Please enter correct choice  ");
            break;
        }
    }
    return 0;
}

/* Create an empty queue */
void create()
{
    front = rear = NULL;
}

/* Returns queue size */
void queuesize()
{
    printf("\n Queue size : %d", count);
}

/* Enqueing the queue */
void enq(int data)
{
    if (rear == NULL)
    {
        rear = (struct node *)malloc(1*sizeof(struct node));
```

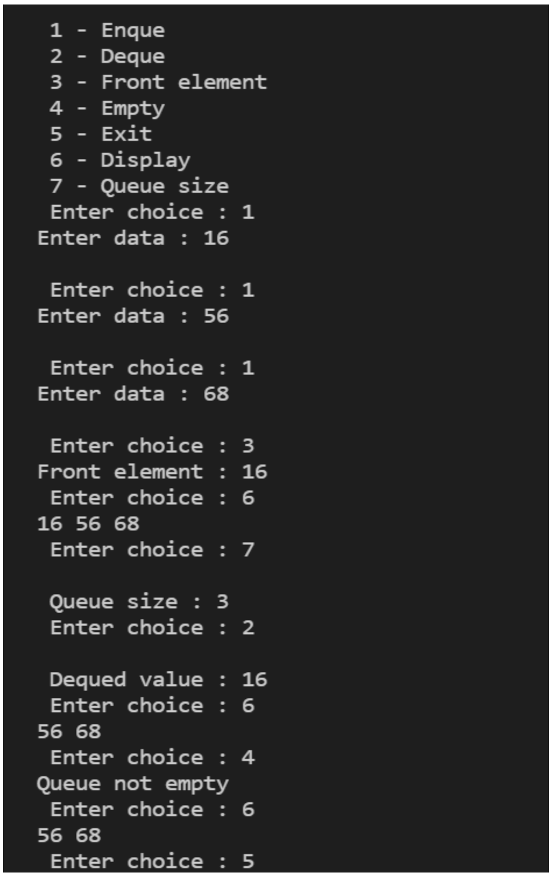


```
        rear->ptr = NULL;
        rear->info = data;
        front = rear;
    }
    else
    {
        temp=(struct node *)malloc(1*sizeof(struct node));
        rear->ptr = temp;
        temp->info = data;
        temp->ptr = NULL;

        rear = temp;
    }
    count++;
}

/* Displaying the queue elements */
void display()
{
    front1 = front;

    if ((front1 == NULL) && (rear == NULL))
    {
        printf("Queue is empty");
        return;
    }
    while (front1 != rear)
    {
        printf("%d ", front1->info);
        front1 = front1->ptr;
    }
    if (front1 == rear)
        printf("%d", front1->info);
}

/* Dequeing the queue */
void deq()
{
    front1 = front;

    if (front1 == NULL)
    {
        printf("\n Error: Trying to display elements from empty queue");
return;
    }
    else
        if (front1->ptr != NULL)
        {
            front1 = front1->ptr;
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = front1;
        }
        else
        {
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = NULL;
            rear = NULL;
        }
        count--;
}
```



```
/* Returns the front element of queue */
int frontelement()
{
    if ((front != NULL) && (rear != NULL))
        return(front->info);
    else
        return 0;
}

/* Display if queue is empty or not */
void empty()
{
     if ((front == NULL) && (rear == NULL))
        printf("\n Queue empty");
    else
       printf("Queue not empty");
}
```

Output:



```
 1 - Enque
 2 - Deque
 3 - Front element
 4 - Empty
 5 - Exit
 6 - Display
 7 - Queue size
 Enter choice : 1
Enter data : 16

 Enter choice : 1
Enter data : 56

 Enter choice : 1
Enter data : 68

 Enter choice : 3
Front element : 16
 Enter choice : 6
16 56 68
 Enter choice : 7

 Queue size : 3
 Enter choice : 2

 Dequed value : 16
 Enter choice : 6
56 68
 Enter choice : 4
Queue not empty
 Enter choice : 6
56 68
 Enter choice : 5
```