Practical no: 12

Objective: Write a program in C to implement circular queue using

1. Array
2. Linked List

Program Codes: Following is the code of this problems in C:-

1. Practical12a.c

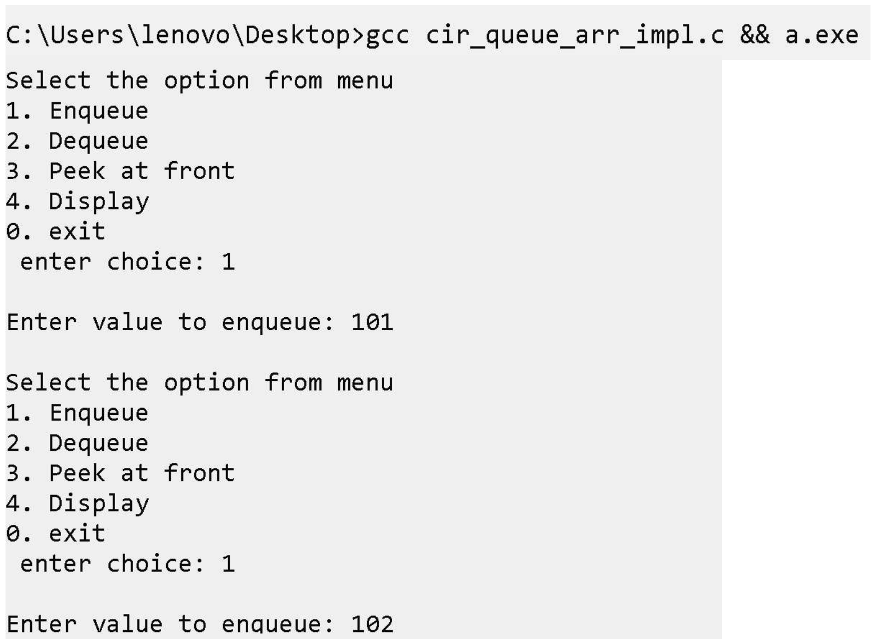
```
#include <stdio.h>
#define MAX 6
int queue[MAX], front = -1, rear = -1, choice = 0;

void enqueue() {
    int element;
    printf("\nEnter value to enqueue: ");
    scanf("%d", &element);
    if ((rear + 1) % MAX == front) {
        printf("\nQueue is full");
        return;
    }
    if (front == -1) {
        front = 0;
    }
    rear = (rear + 1) % MAX; // Circular increment
    queue[rear] = element;
}

void dequeue() {
    if (front == -1) {
        printf("\nQueue is empty");
        return;
    }
    printf("\nDequeued value: %d", queue[front]);
    if (front == rear) {
        front = rear = -1;
    } else {
        front = (front + 1) % MAX; // Circular increment
    }
}
void value_at_front() {
    if (front == -1) {
        printf("\nQueue is empty");
        return;
    }
    printf("\nValue at front is: %d", queue[front]);
}
void display() {
    if (front == -1) {
        printf("\nQueue is empty");
        return;
    }
    printf("\nValues are: ");
    int i = front;
    while (1) {
        printf("%5d", queue[i]);
        if (i == rear) break;
        i = (i + 1) % MAX; // Circular increment
    }
}

int main() {
    while (1) {
        printf("\nSelect the option from menu"
               "\n1. Enqueue"
               "\n2. Dequeue"
               "\n3. Peek at front"
               "\n4. Display"
               "\n0. exit"
               "\nEnter choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: enqueue(); break;
            case 2: dequeue(); break;
            case 3: value_at_front(); break;
            case 4: display(); break;
            case 0: return 0;
            default: printf("\nWrong choice, retry");
        }
    }
}
```

2. Practical12b.c

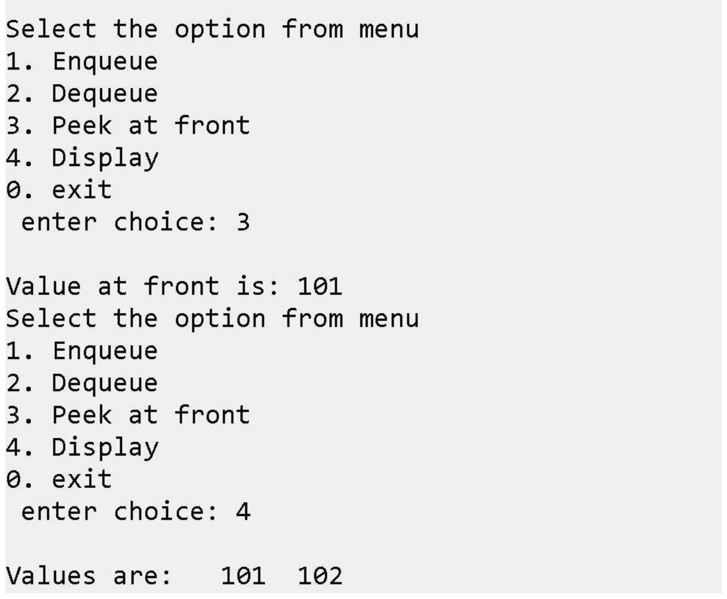
```c
#include <stdlib.h>
#include <stdio.h>
typedef struct node {
    int data;
    struct node *link;
} node;
node *head = NULL, *tail = NULL;

void enqueue() {
    node *new_node = (node *)malloc(sizeof(node));
    if (!new_node) {
        printf("Memory allocation failed\n");
        return;
    }
    printf("Enter data to enqueue: ");
    scanf("%d", &new_node->data);
    if (head == NULL) {
        head = tail = new_node;
        tail->link = head; // Point tail to head to make it circular
    } else {
        tail->link = new_node; // Link the new node
        tail = new_node;        // Update tail to the new node
        tail->link = head;      // Maintain circular link
    }
}

void dequeue() {
    if (head == NULL) {
        printf("Queue is empty\n");
        return;
    }
    node *temp = head;
    if (head == tail) {
        head = tail = NULL;
    } else {
        head = head->link; // Move head to the next node
        tail->link = head; // Update tail's link to new head
    }
    printf("Dequeued data is: %d\n", temp->data);
    free(temp);
}
```

```c
void value_at_front() {
    if (head == NULL) {
        printf("Queue is empty\n");
        return;
    }
    printf("Value at front is: %d\n", head->data);
}

void display() {
    if (head == NULL) {
        printf("Queue is empty\n");
        return;
    }
    node *temp = head;
    do {
        printf("%5d", temp->data);
        temp = temp->link;
    } while (temp != head);
    printf("\n");
}

int main() {
    int choice;
    while (1) {
        printf("\nSelect the option from menu"
               "\n1. Enqueue"
               "\n2. Dequeue"
               "\n3. Peek at front"
               "\n4. Display"
               "\n0. exit"
               "\nEnter choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: enqueue(); break;
            case 2: dequeue(); break;
            case 3: value_at_front(); break;
            case 4: display(); break;
            case 0: return 0;
            default: printf("Wrong choice, retry\n");
        }
    }}
```

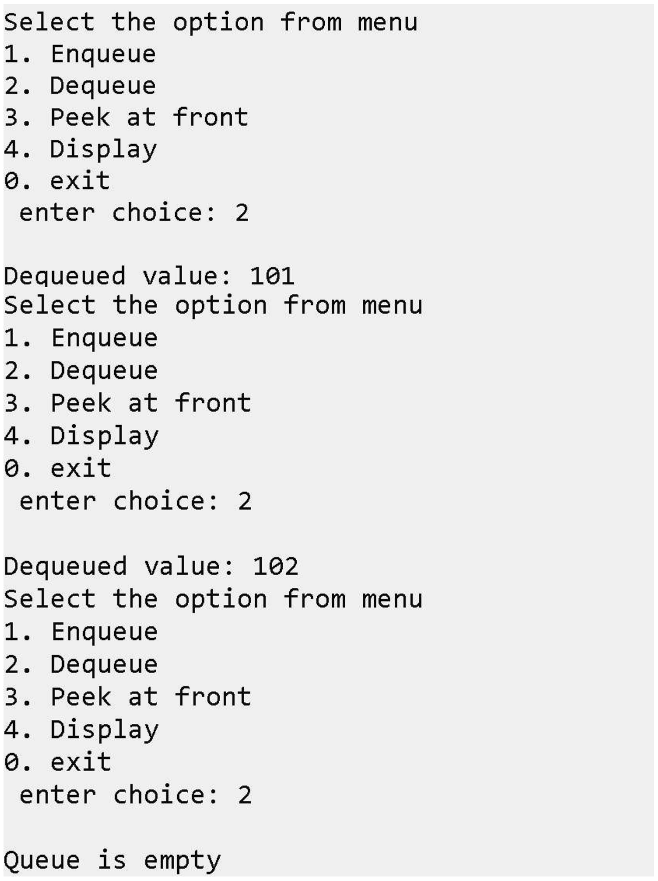
Output: Following is the output of the program:-



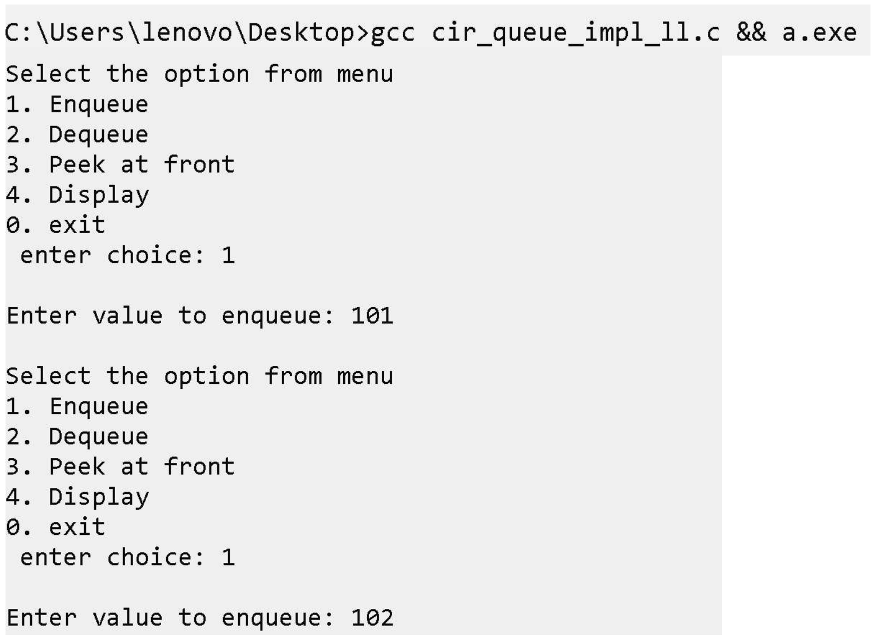
```
C:\Users\lenovo\Desktop>gcc cir_queue_arr_impl.c && a.exe
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 1

Enter value to enqueue: 101

Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 1

Enter value to enqueue: 102
```



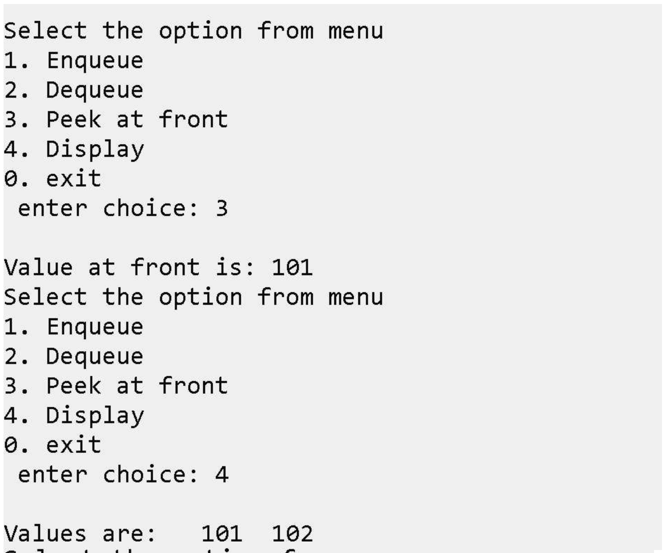
```
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 3

Value at front is: 101
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 4

Values are:   101  102
```



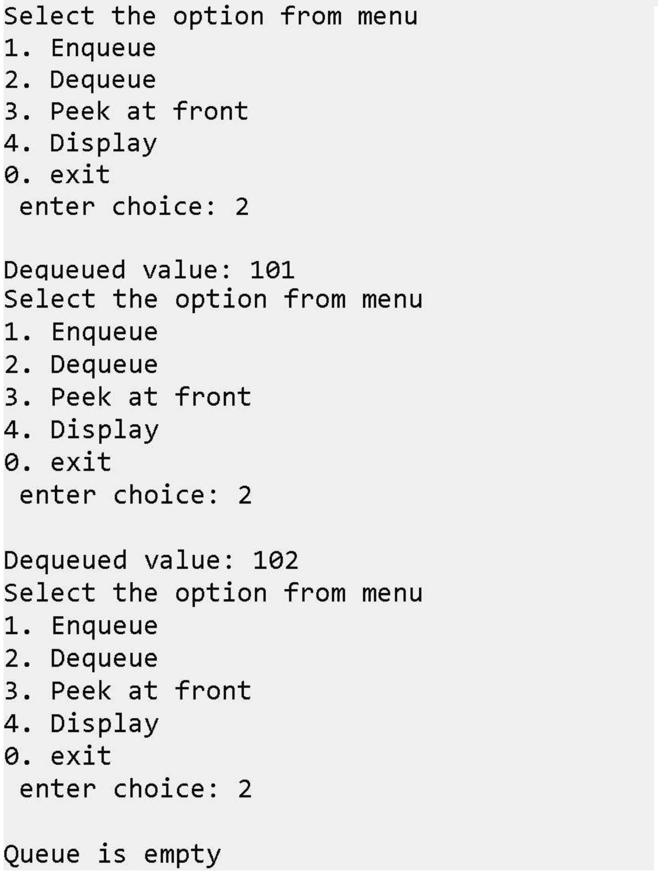
```
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 2

Dequeued value: 101
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 2

Dequeued value: 102
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 2

Queue is empty
```



```
C:\Users\lenovo\Desktop>gcc cir_queue_impl_ll.c && a.exe
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 1

Enter value to enqueue: 101

Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 1

Enter value to enqueue: 102
```



```
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 3

Value at front is: 101
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 4

Values are:   101  102
```



```
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 2

Dequeued value: 101
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 2

Dequeued value: 102
Select the option from menu
1. Enqueue
2. Dequeue
3. Peek at front
4. Display
0. exit
 enter choice: 2

Queue is empty
```