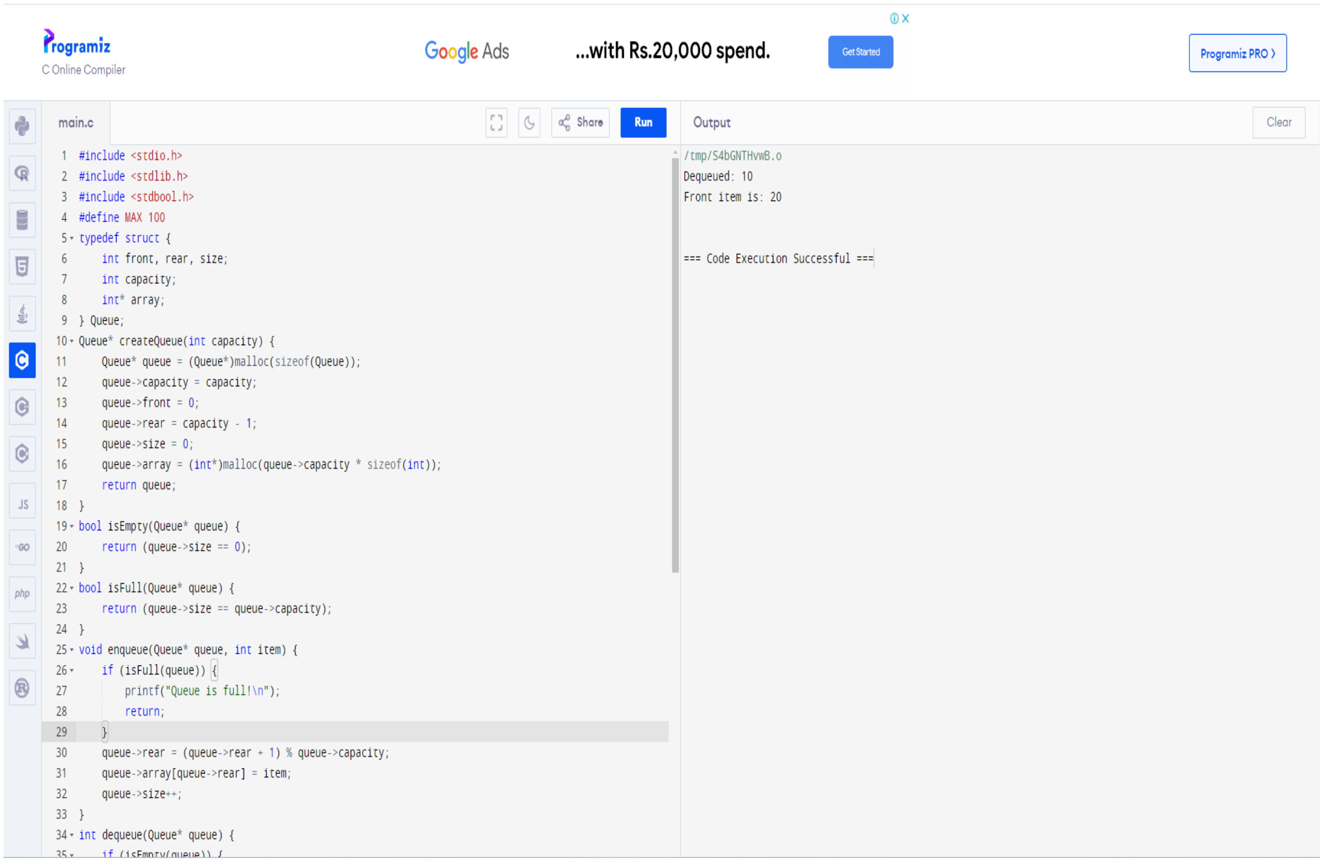
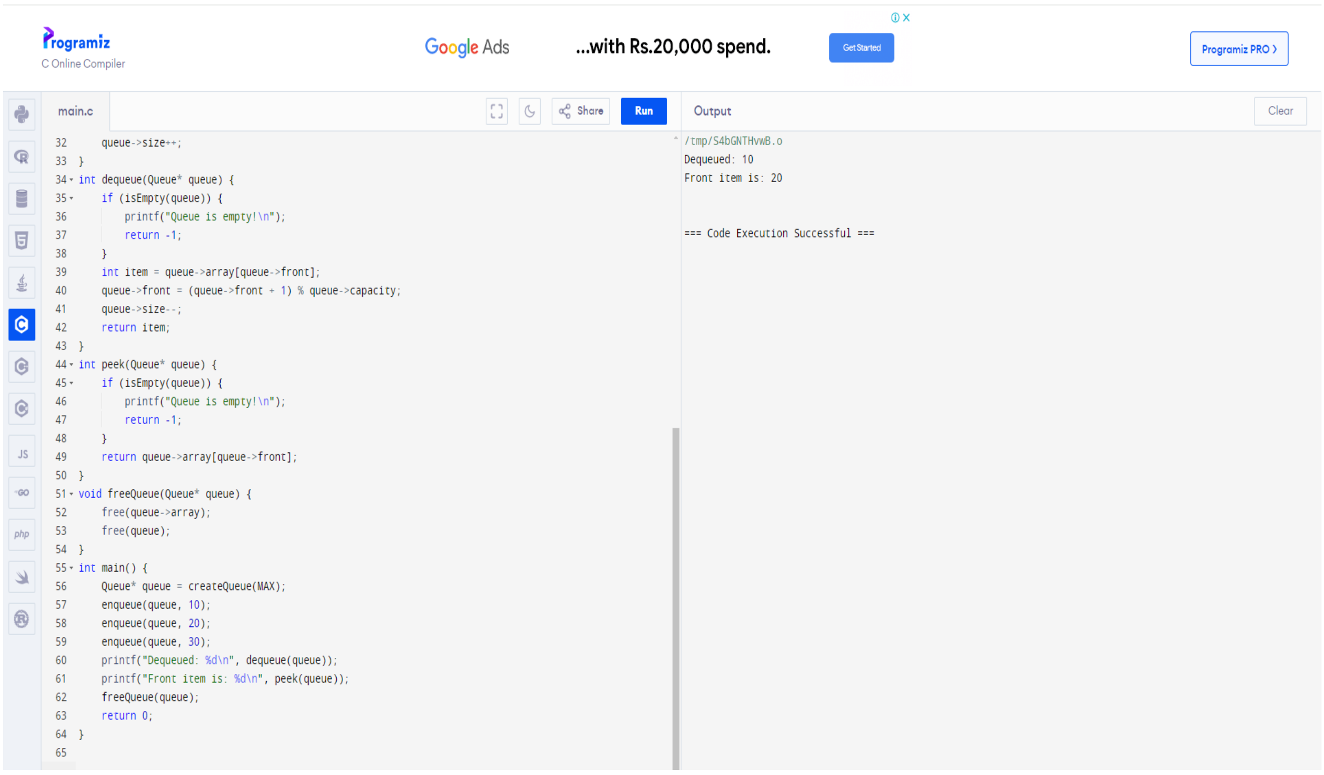
1.Write a c program for Implementation of array in Queue.



```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 100
typedef struct {
    int front, rear, size;
    int capacity;
    int* array;
} Queue;
Queue* createQueue(int capacity) {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->capacity = capacity;
    queue->front = 0;
    queue->rear = capacity - 1;
    queue->size = 0;
    queue->array = (int*)malloc(queue->capacity * sizeof(int));
    return queue;
}
bool isEmpty(Queue* queue) {
    return (queue->size == 0);
}
bool isFull(Queue* queue) {
    return (queue->size == queue->capacity);
}
void enqueue(Queue* queue, int item) {
    if (isFull(queue)) {
        printf("Queue is full!\n");
        return;
    }
    queue->rear = (queue->rear + 1) % queue->capacity;
    queue->array[queue->rear] = item;
    queue->size++;
}
int dequeue(Queue* queue) {
    if (isEmpty(queue)) {
```

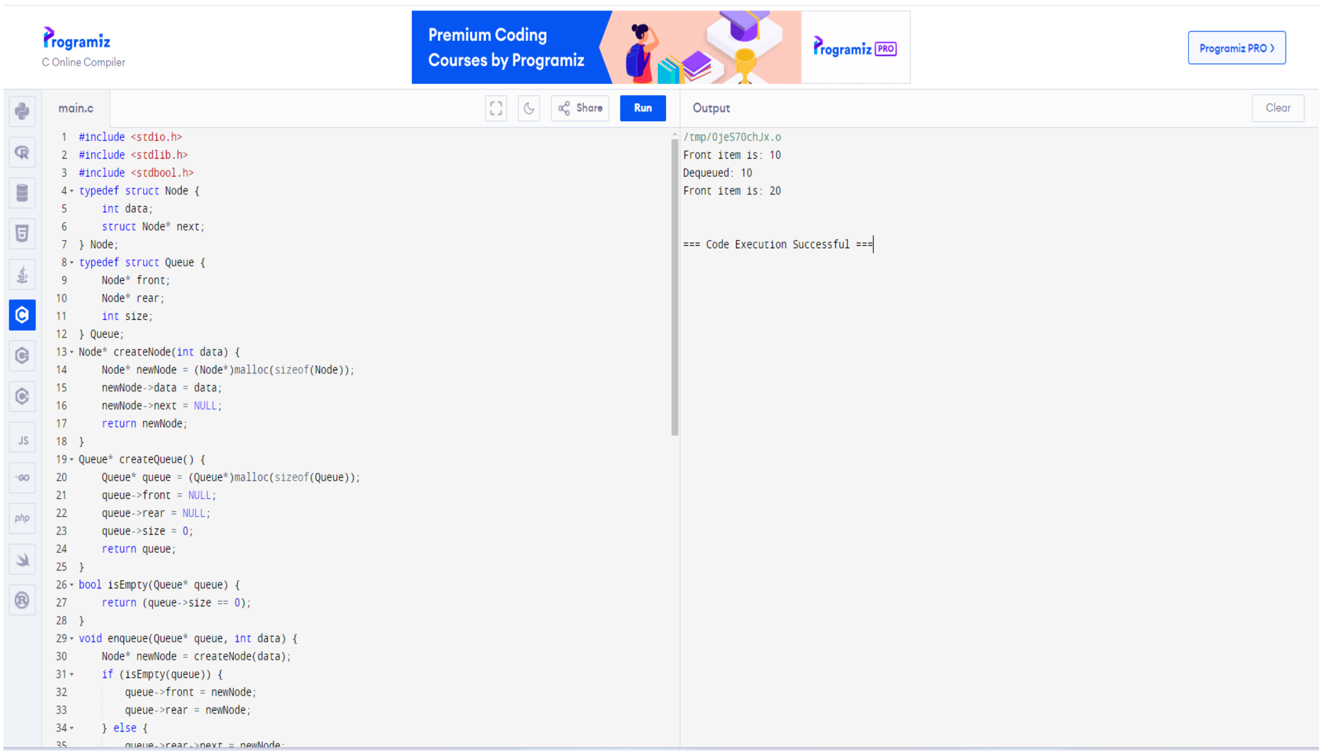
```
/tmp/S4bGNTHvwB.o
Dequeued: 10
Front item is: 20


=== Code Execution Successful ===
```



```c
    queue->size++;
}
int dequeue(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty!\n");
        return -1;
    }
    int item = queue->array[queue->front];
    queue->front = (queue->front + 1) % queue->capacity;
    queue->size--;
    return item;
}
int peek(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty!\n");
        return -1;
    }
    return queue->array[queue->front];
}
void freeQueue(Queue* queue) {
    free(queue->array);
    free(queue);
}
int main() {
    Queue* queue = createQueue(MAX);
    enqueue(queue, 10);
    enqueue(queue, 20);
    enqueue(queue, 30);
    printf("Dequeued: %d\n", dequeue(queue));
    printf("Front item is: %d\n", peek(queue));
    freeQueue(queue);
    return 0;
}
```

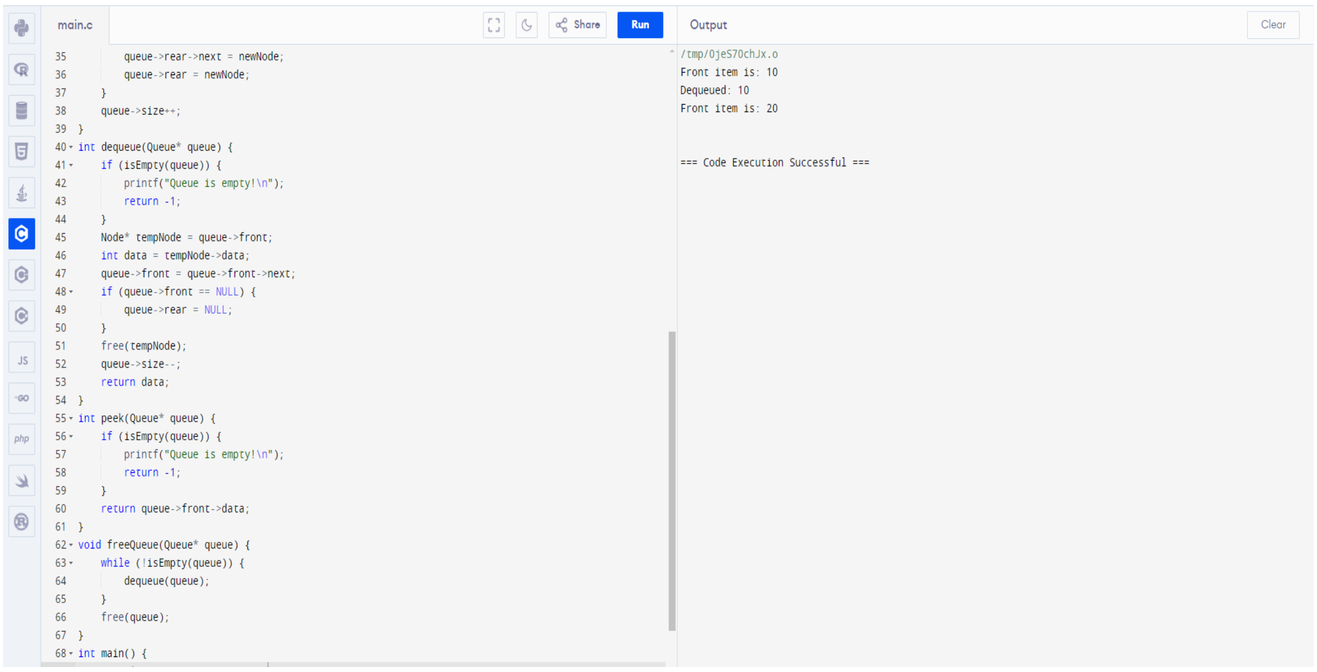
```
/tmp/S4bGNTHvwB.o
Dequeued: 10
Front item is: 20

=== Code Execution Successful ===
```

2.Write a c program for Implementation of Linked List in Queue.



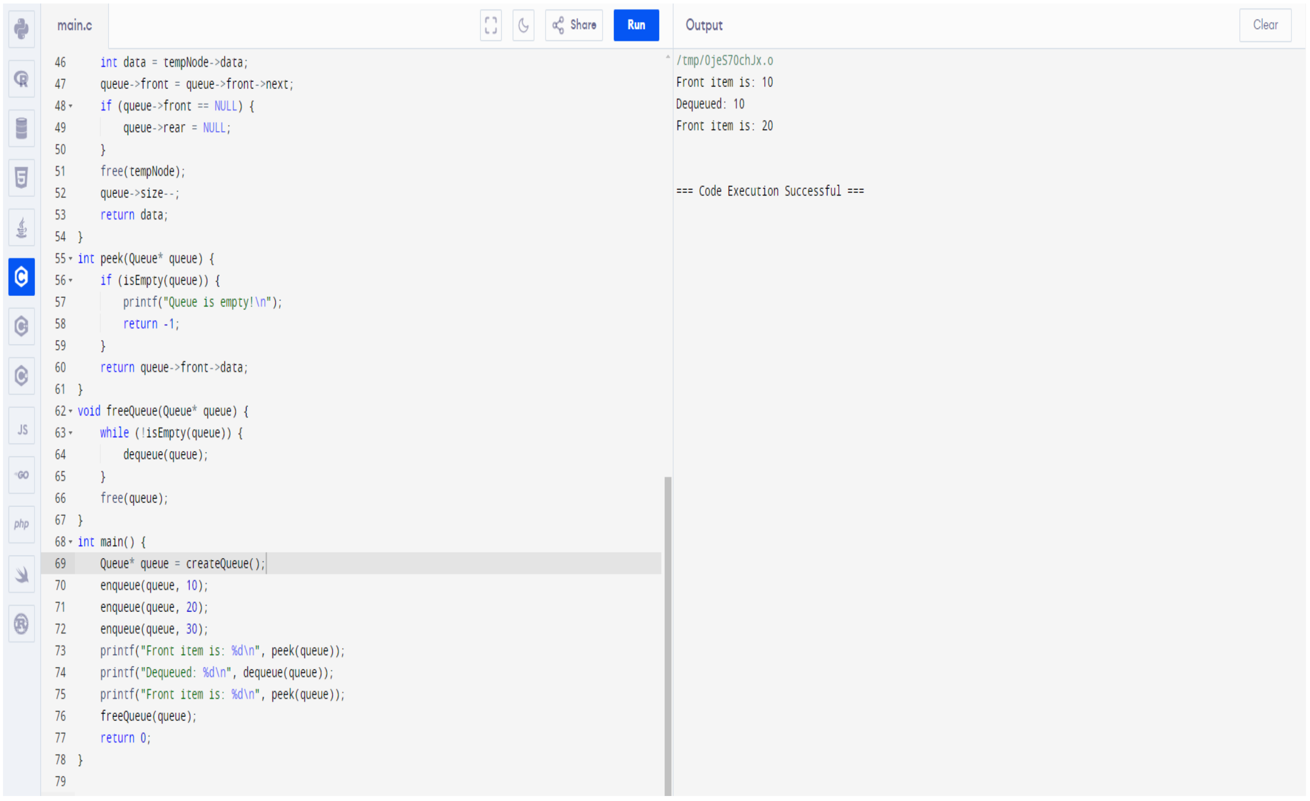
```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
typedef struct Queue {
    Node* front;
    Node* rear;
    int size;
} Queue;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
Queue* createQueue() {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->front = NULL;
    queue->rear = NULL;
    queue->size = 0;
    return queue;
}
bool isEmpty(Queue* queue) {
    return (queue->size == 0);
}
void enqueue(Queue* queue, int data) {
    Node* newNode = createNode(data);
    if (isEmpty(queue)) {
        queue->front = newNode;
        queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
```

```
/tmp/0jeS70chJx.o
Front item is: 10
Dequeued: 10
Front item is: 20


=== Code Execution Successful ===
```



```c
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
    queue->size++;
}
int dequeue(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty!\n");
        return -1;
    }
    Node* tempNode = queue->front;
    int data = tempNode->data;
    queue->front = queue->front->next;
    if (queue->front == NULL) {
        queue->rear = NULL;
    }
    free(tempNode);
    queue->size--;
    return data;
}
int peek(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty!\n");
        return -1;
    }
    return queue->front->data;
}
void freeQueue(Queue* queue) {
    while (!isEmpty(queue)) {
        dequeue(queue);
    }
    free(queue);
}
int main() {
```

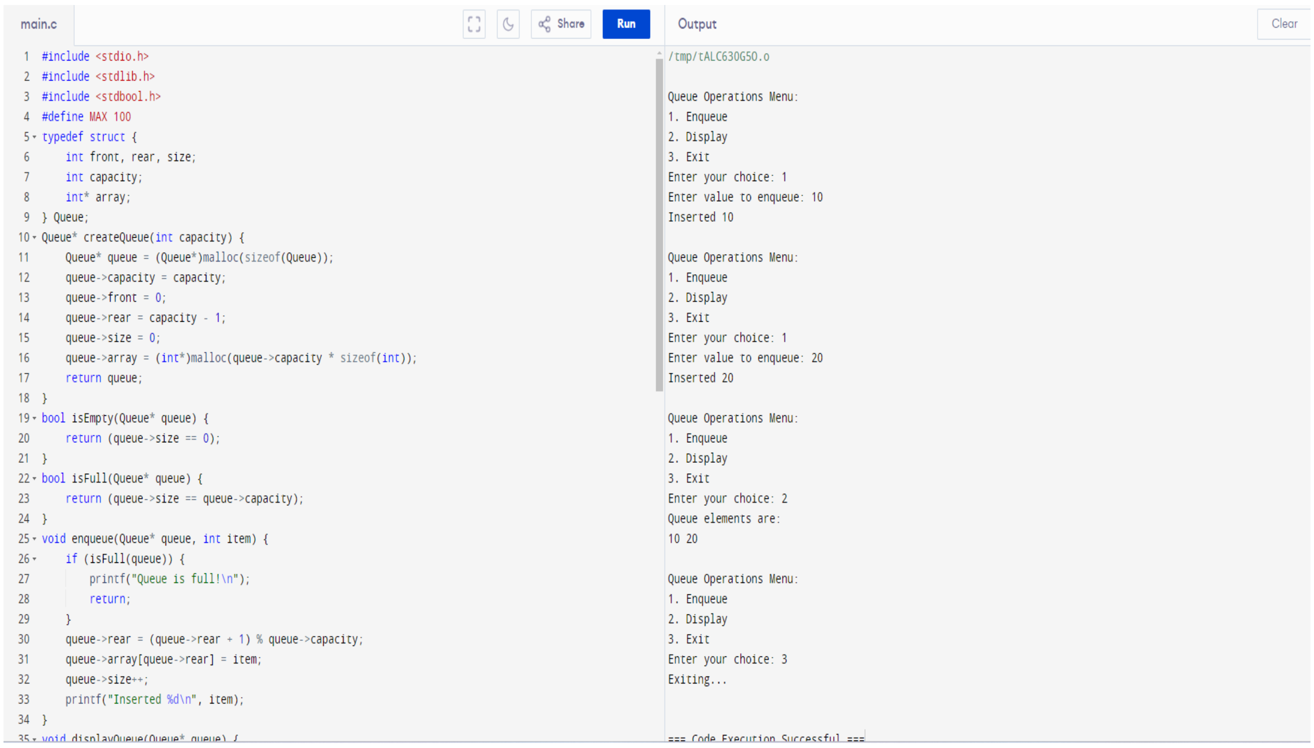
```
/tmp/0jeS70chJx.o
Front item is: 10
Dequeued: 10
Front item is: 20


=== Code Execution Successful ===
```



```c
    int data = tempNode->data;
    queue->front = queue->front->next;
    if (queue->front == NULL) {
        queue->rear = NULL;
    }
    free(tempNode);
    queue->size--;
    return data;
}
int peek(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty!\n");
        return -1;
    }
    return queue->front->data;
}
void freeQueue(Queue* queue) {
    while (!isEmpty(queue)) {
        dequeue(queue);
    }
    free(queue);
}
int main() {
    Queue* queue = createQueue();
    enqueue(queue, 10);
    enqueue(queue, 20);
    enqueue(queue, 30);
    printf("Front item is: %d\n", peek(queue));
    printf("Dequeued: %d\n", dequeue(queue));
    printf("Front item is: %d\n", peek(queue));
    freeQueue(queue);
    return 0;
}
```

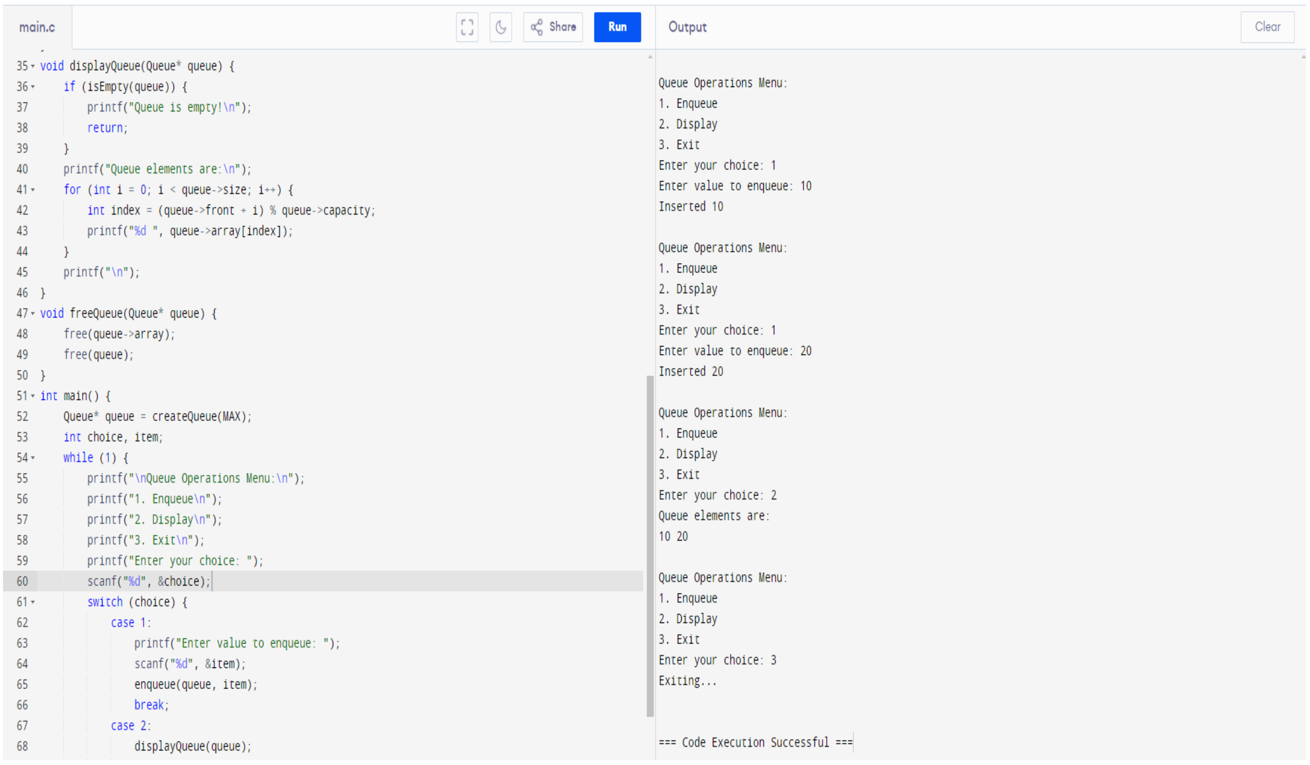
```
/tmp/0jeS70chJx.o
Front item is: 10
Dequeued: 10
Front item is: 20


=== Code Execution Successful ===
```

3.Write a c program for Queue Operation of Enqueue and display.



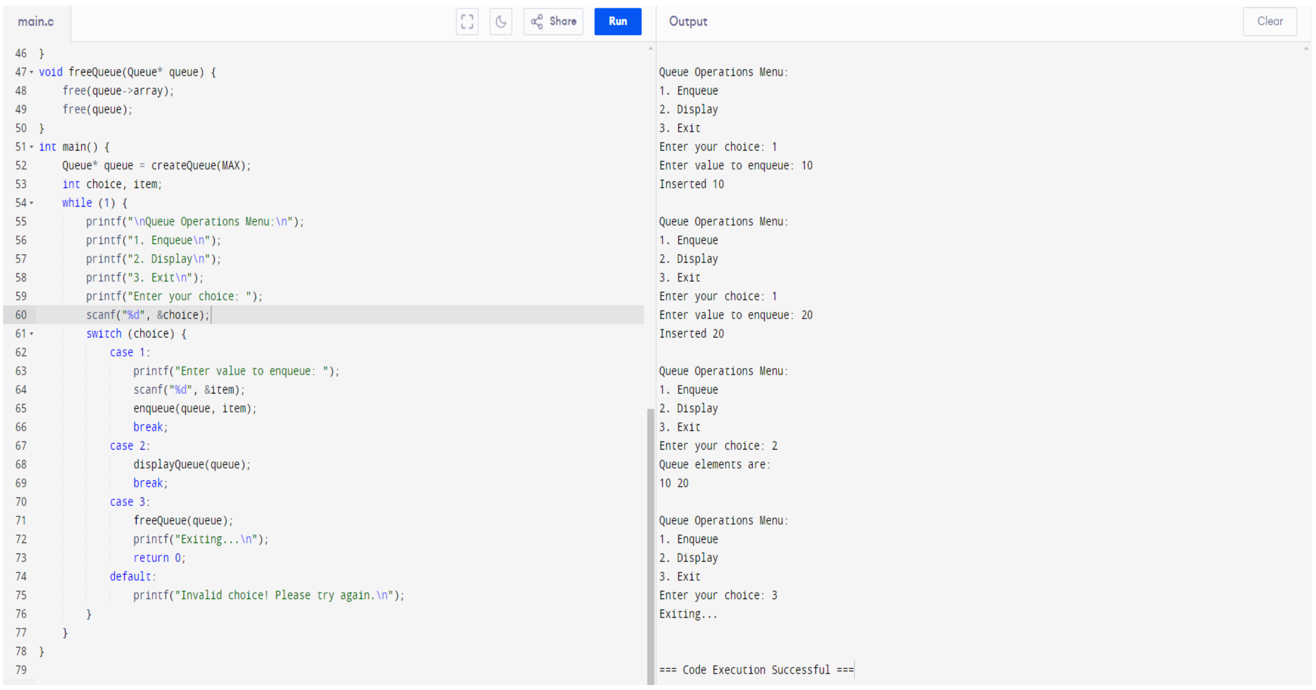
```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 100
typedef struct {
    int front, rear, size;
    int capacity;
    int* array;
} Queue;
Queue* createQueue(int capacity) {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->capacity = capacity;
    queue->front = 0;
    queue->rear = capacity - 1;
    queue->size = 0;
    queue->array = (int*)malloc(queue->capacity * sizeof(int));
    return queue;
}
bool isEmpty(Queue* queue) {
    return (queue->size == 0);
}
bool isFull(Queue* queue) {
    return (queue->size == queue->capacity);
}
void enqueue(Queue* queue, int item) {
    if (isFull(queue)) {
        printf("Queue is full!\n");
        return;
    }
    queue->rear = (queue->rear + 1) % queue->capacity;
    queue->array[queue->rear] = item;
    queue->size++;
    printf("Inserted %d\n", item);
}
void displayQueue(Queue* queue) {
```

```
/tmp/tALC630G50.o

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 10
Inserted 10

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 20
Inserted 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 2
Queue elements are:
10 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 3
Exiting...


=== Code Execution Successful ===
```



```c
void displayQueue(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty!\n");
        return;
    }
    printf("Queue elements are:\n");
    for (int i = 0; i < queue->size; i++) {
        int index = (queue->front + i) % queue->capacity;
        printf("%d ", queue->array[index]);
    }
    printf("\n");
}
void freeQueue(Queue* queue) {
    free(queue->array);
    free(queue);
}
int main() {
    Queue* queue = createQueue(MAX);
    int choice, item;
    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Display\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &item);
                enqueue(queue, item);
                break;
            case 2:
                displayQueue(queue);
```

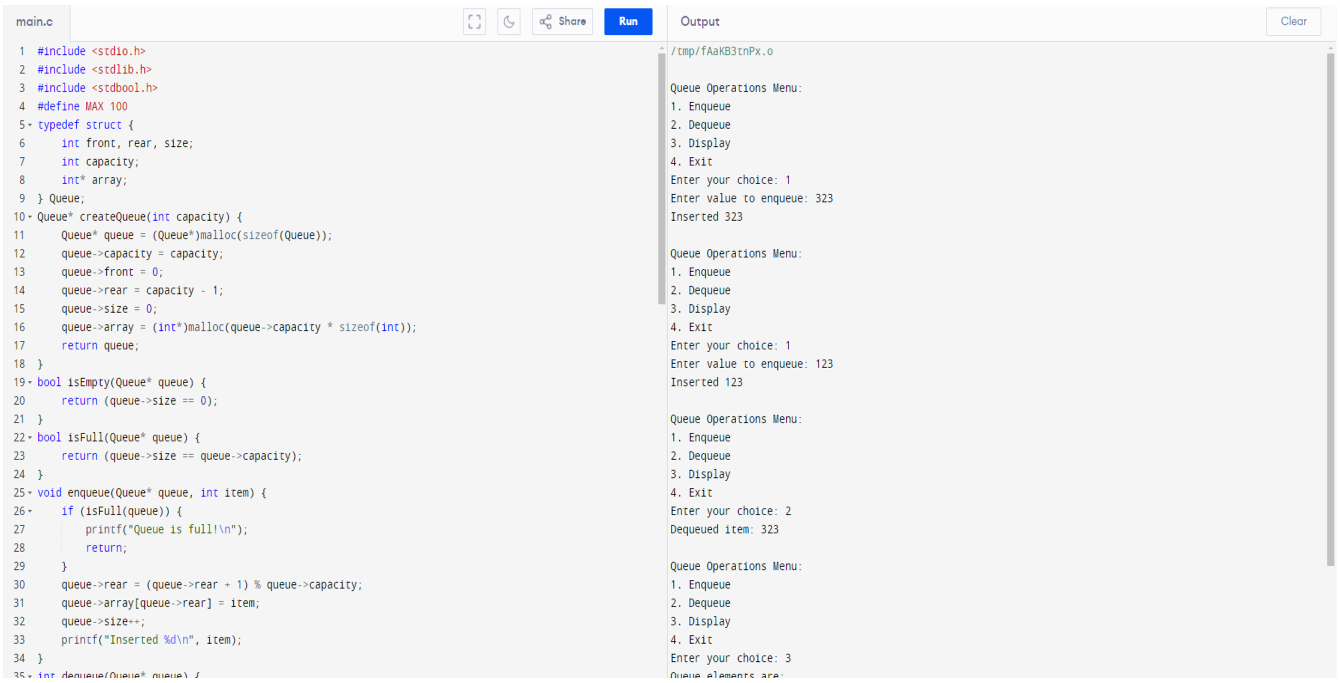
```

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 10
Inserted 10

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 20
Inserted 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 2
Queue elements are:
10 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 3
Exiting...


=== Code Execution Successful ===
```



```c
}
void freeQueue(Queue* queue) {
    free(queue->array);
    free(queue);
}
int main() {
    Queue* queue = createQueue(MAX);
    int choice, item;
    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Display\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &item);
                enqueue(queue, item);
                break;
            case 2:
                displayQueue(queue);
                break;
            case 3:
                freeQueue(queue);
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
}
```

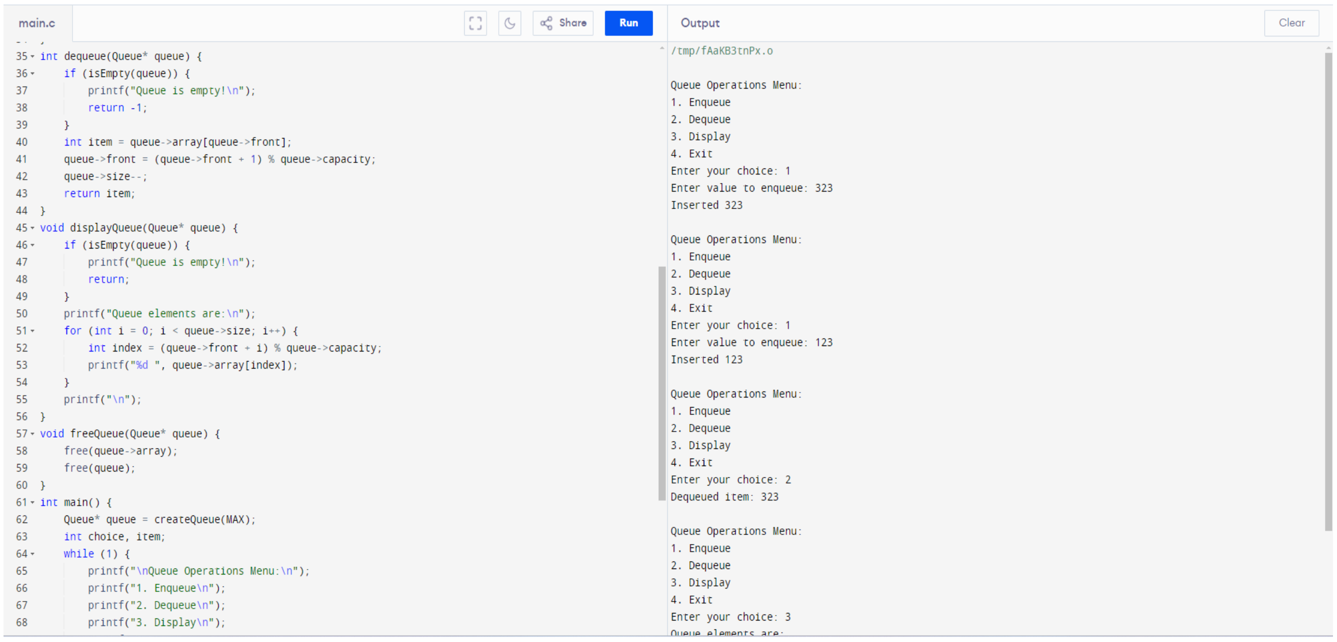
```
Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 10
Inserted 10

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 20
Inserted 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 2
Queue elements are:
10 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 3
Exiting...


=== Code Execution Successful ===
```

4.Write a c program for Queue operation of Dequeue and display.



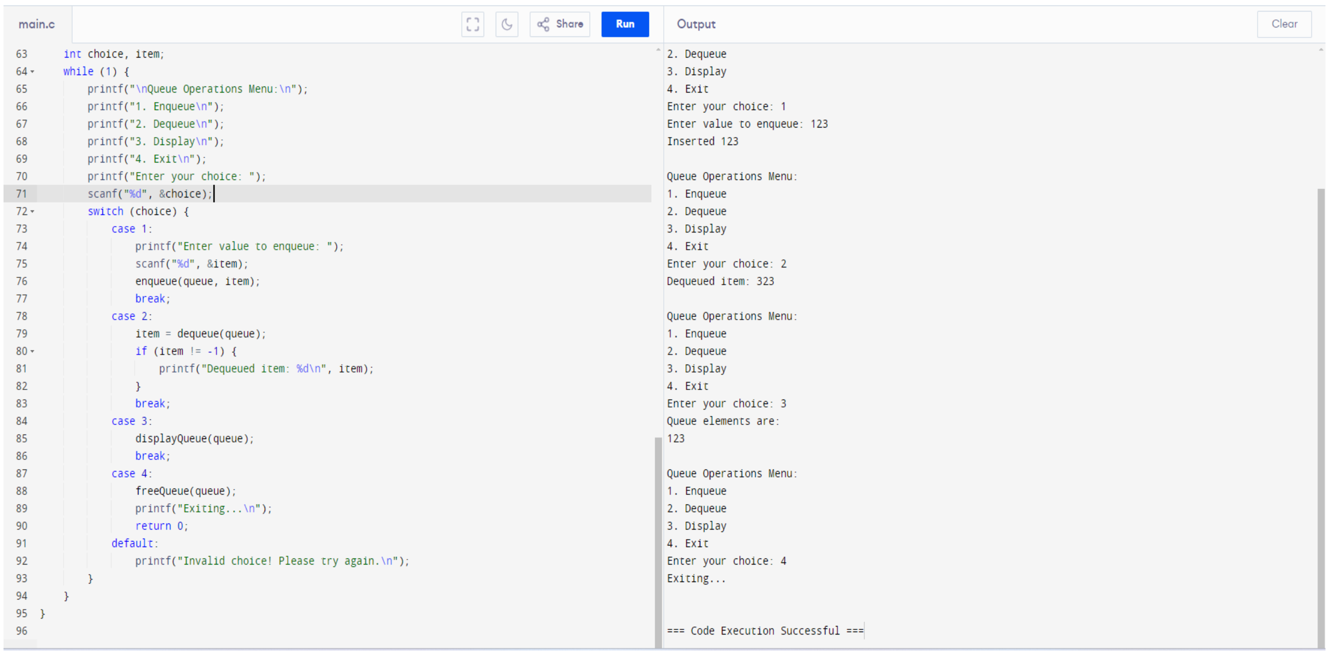
```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 100
typedef struct {
    int front, rear, size;
    int capacity;
    int* array;
} Queue;
Queue* createQueue(int capacity) {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->capacity = capacity;
    queue->front = 0;
    queue->rear = capacity - 1;
    queue->size = 0;
    queue->array = (int*)malloc(queue->capacity * sizeof(int));
    return queue;
}
bool isEmpty(Queue* queue) {
    return (queue->size == 0);
}
bool isFull(Queue* queue) {
    return (queue->size == queue->capacity);
}
void enqueue(Queue* queue, int item) {
    if (isFull(queue)) {
        printf("Queue is full!\n");
        return;
    }
    queue->rear = (queue->rear + 1) % queue->capacity;
    queue->array[queue->rear] = item;
    queue->size++;
    printf("Inserted %d\n", item);
}
```

```
/tmp/fAaKB3tnPx.o

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 323
Inserted 323

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 123
Inserted 123

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued item: 323

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
```



```c
int dequeue(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty!\n");
        return -1;
    }
    int item = queue->array[queue->front];
    queue->front = (queue->front + 1) % queue->capacity;
    queue->size--;
    return item;
}
void displayQueue(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty!\n");
        return;
    }
    printf("Queue elements are:\n");
    for (int i = 0; i < queue->size; i++) {
        int index = (queue->front + i) % queue->capacity;
        printf("%d ", queue->array[index]);
    }
    printf("\n");
}
void freeQueue(Queue* queue) {
    free(queue->array);
    free(queue);
}
int main() {
    Queue* queue = createQueue(MAX);
    int choice, item;
    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
```

```
/tmp/fAaKB3tnPx.o

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 323
Inserted 323

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 123
Inserted 123

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued item: 323

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
```



```c
    int choice, item;
    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &item);
                enqueue(queue, item);
                break;
            case 2:
                item = dequeue(queue);
                if (item != -1) {
                    printf("Dequeued item: %d\n", item);
                }
                break;
            case 3:
                displayQueue(queue);
                break;
            case 4:
                freeQueue(queue);
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
}

```

```
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 123
Inserted 123

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued item: 323

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue elements are:
123

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 4
Exiting...


=== Code Execution Successful ===
```