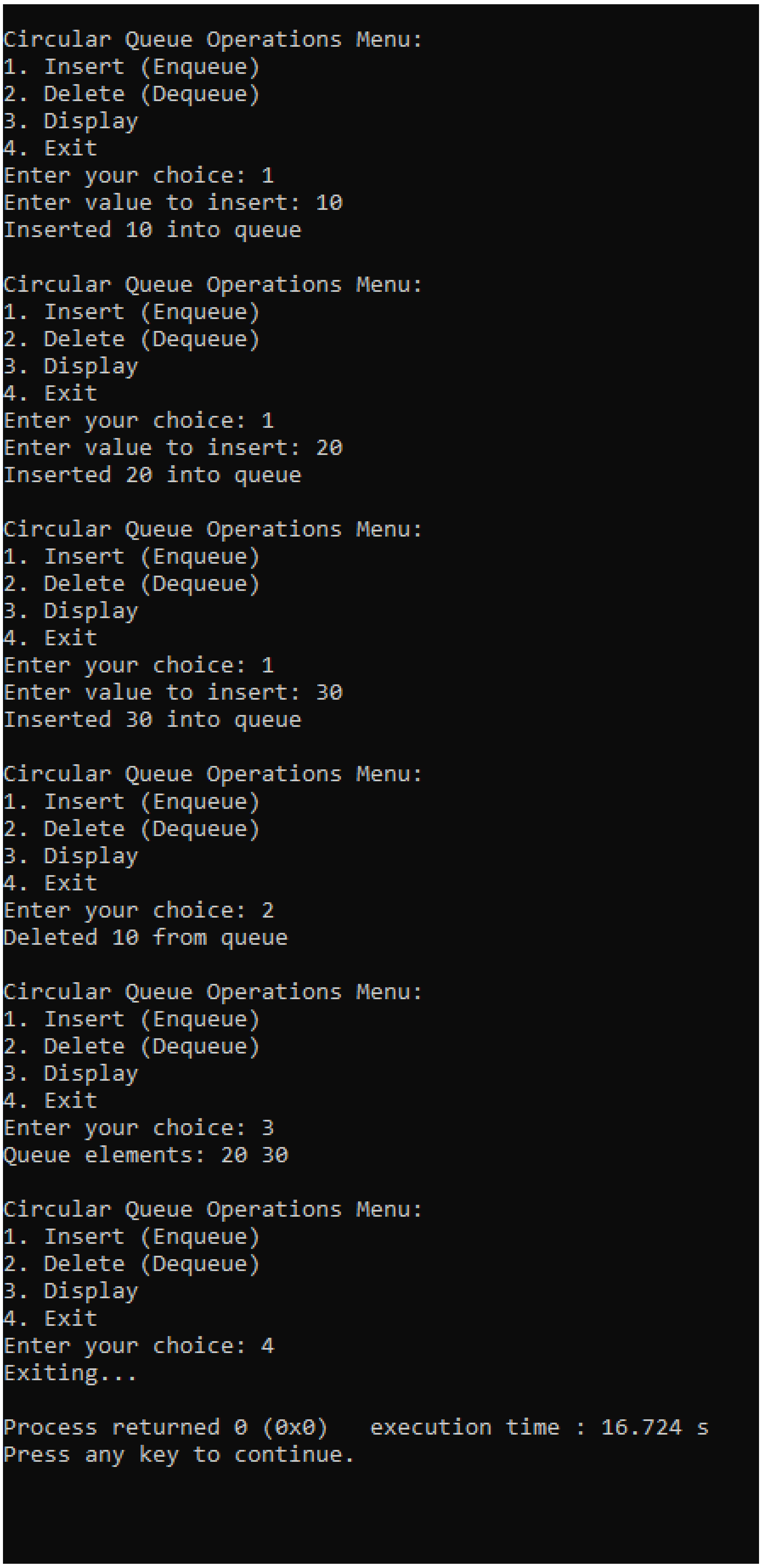
3b. WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: Insert, Delete & Display The program should print appropriate messages for queue empty and queue overflow conditions

```
#include <stdio.h>

#define MAX 5  // Maximum size of the queue


// Declare queue array and front & rear pointers

int queue[MAX];

int front = -1, rear = -1;


// Function to check if the queue is full

int isFull() {

    if ((rear + 1) % MAX == front)

        return 1;  // Queue is full

    return 0;

}


// Function to check if the queue is empty

int isEmpty() {

    if (front == -1)

        return 1;  // Queue is empty

    return 0;

}


// Function to insert an element into the queue

void insert(int value) {

    if (isFull()) {

        printf("Queue Overflow! Cannot insert %d\n", value);

    } else {

        if (front == -1)  // First element to be inserted

            front = 0;

        rear = (rear + 1) % MAX;

        queue[rear] = value;
        printf("Inserted %d into queue\n", value);
    }
}

// Function to delete an element from the queue
int delete() {
    if (isEmpty()) {
        printf("Queue Underflow! No elements to delete\n");
        return -1;
    } else {
        int value = queue[front];
        if (front == rear)  // Only one element left in the queue
            front = rear = -1;  // Reset queue
        else
            front = (front + 1) % MAX;
        return value;
    }
}

// Function to display the elements of the queue
void display() {
    if (isEmpty()) {
        printf("Queue is empty!\n");
    } else {
        printf("Queue elements: ");
        int i = front;
        while (i != rear) {
            printf("%d ", queue[i]);
            i = (i + 1) % MAX;
        }
        printf("%d\n", queue[rear]);  // Print the last element
    }
```

```
}

int main() {
    int choice, value;

    while (1) {
        // Menu
        printf("\nCircular Queue Operations Menu:\n");
        printf("1. Insert (Enqueue)\n");
        printf("2. Delete (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 insert: ");
                scanf("%d", &value);
                insert(value);
                break;

            case 2:
                value = delete();
                if (value != -1) {
                    printf("Deleted %d from queue\n", value);
                }
                break;

            case 3:
                display();
                break;
            case 4:
                printf("Exiting...\n");
                return 0;

            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```


```
Circular Queue Operations Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 10
Inserted 10 into queue

Circular Queue Operations Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 20
Inserted 20 into queue

Circular Queue Operations Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 30
Inserted 30 into queue

Circular Queue Operations Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 2
Deleted 10 from queue

Circular Queue Operations Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 3
Queue elements: 20 30

Circular Queue Operations Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 4
Exiting...

Process returned 0 (0x0)   execution time : 16.724 s
Press any key to continue.
```