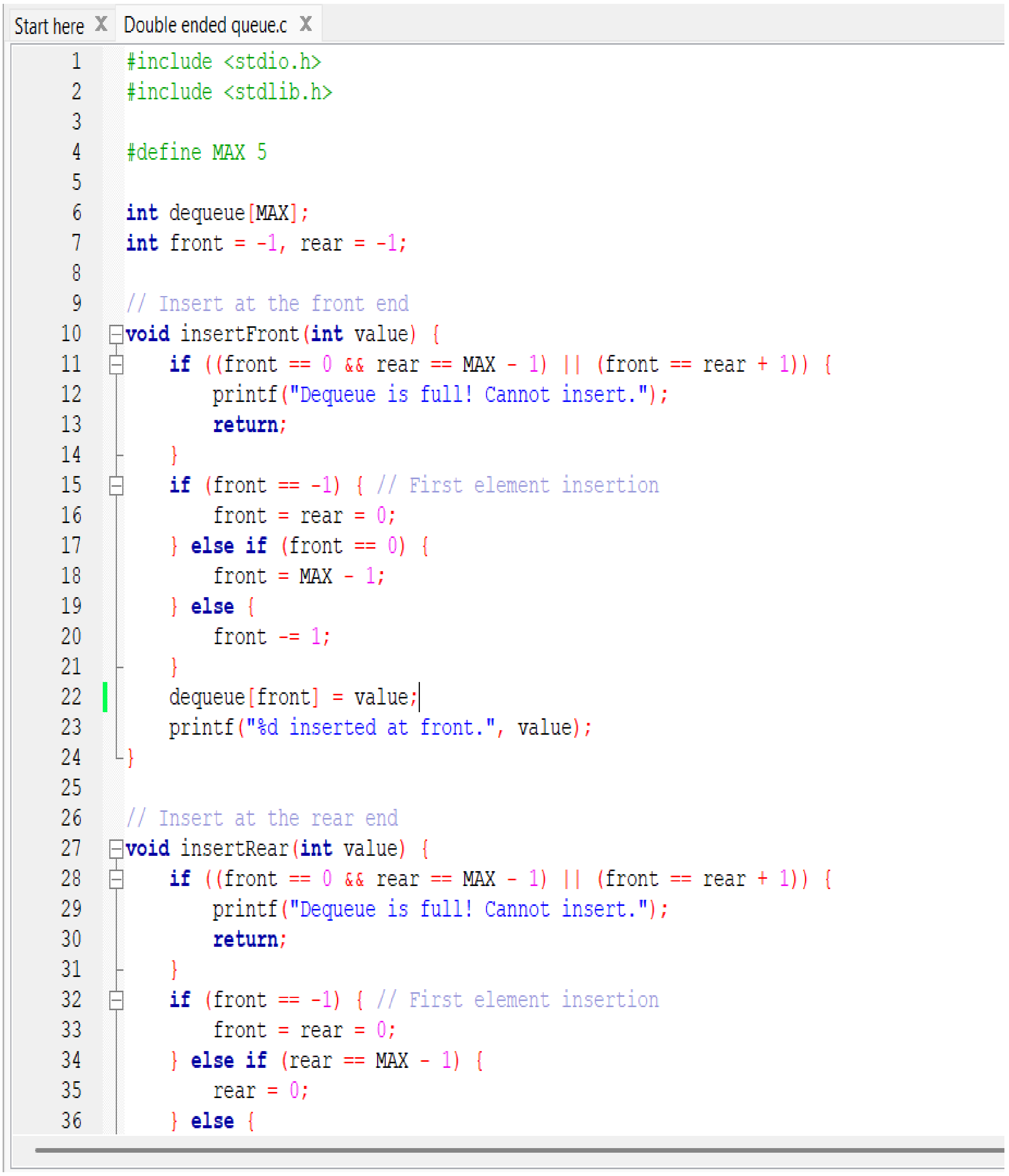
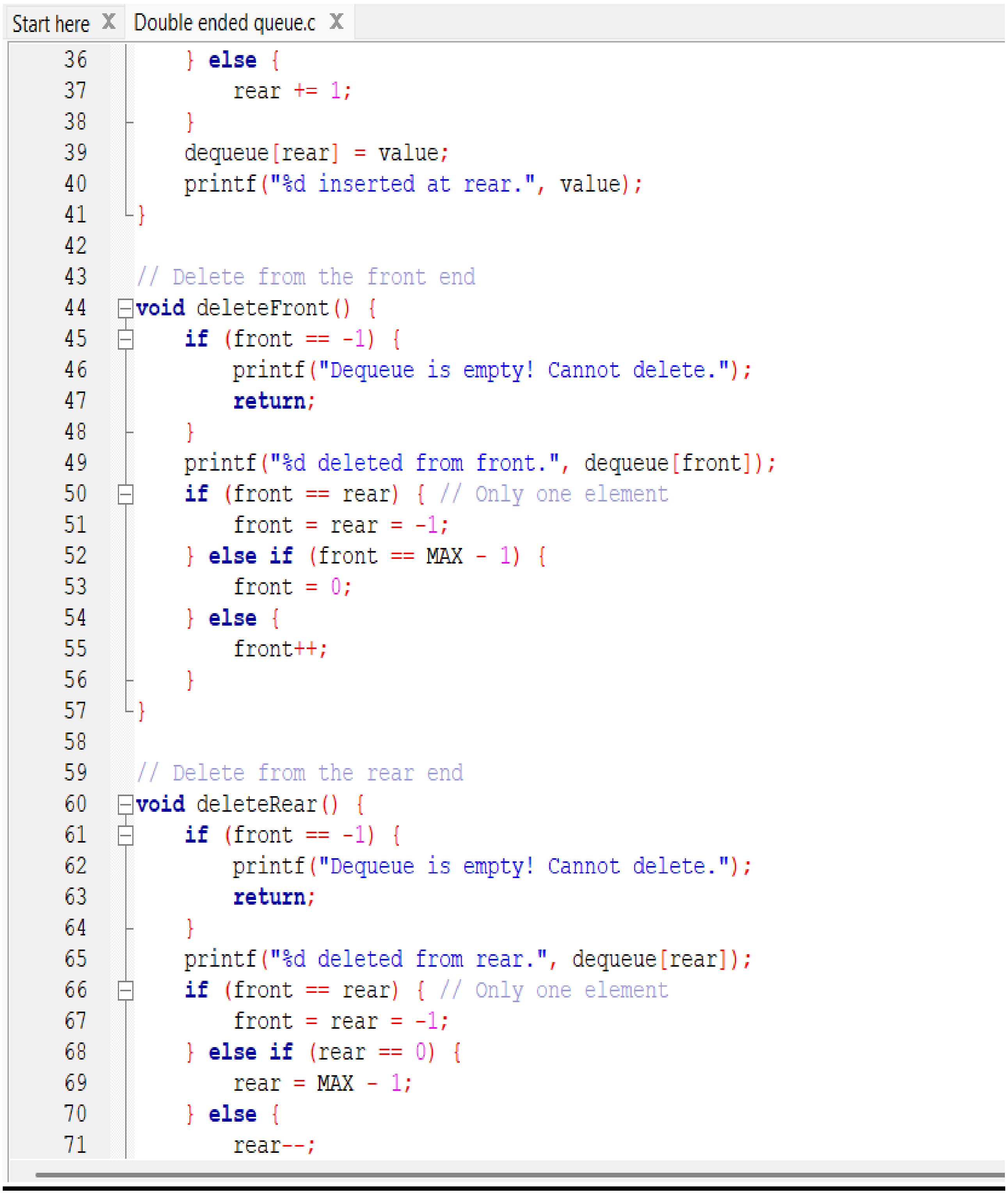
LINKED LIST INSERTION OPERATIONS

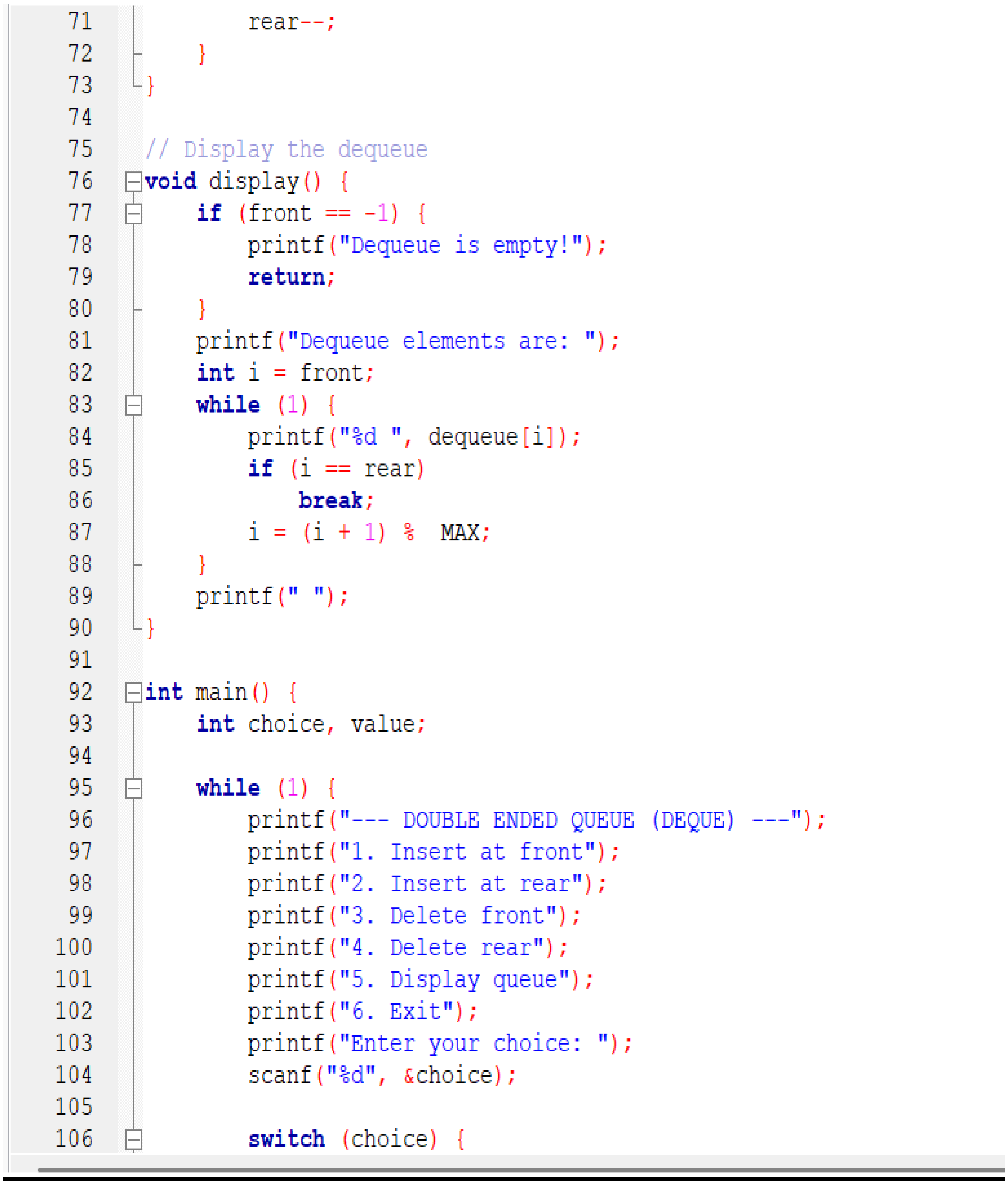
PROGRAM:



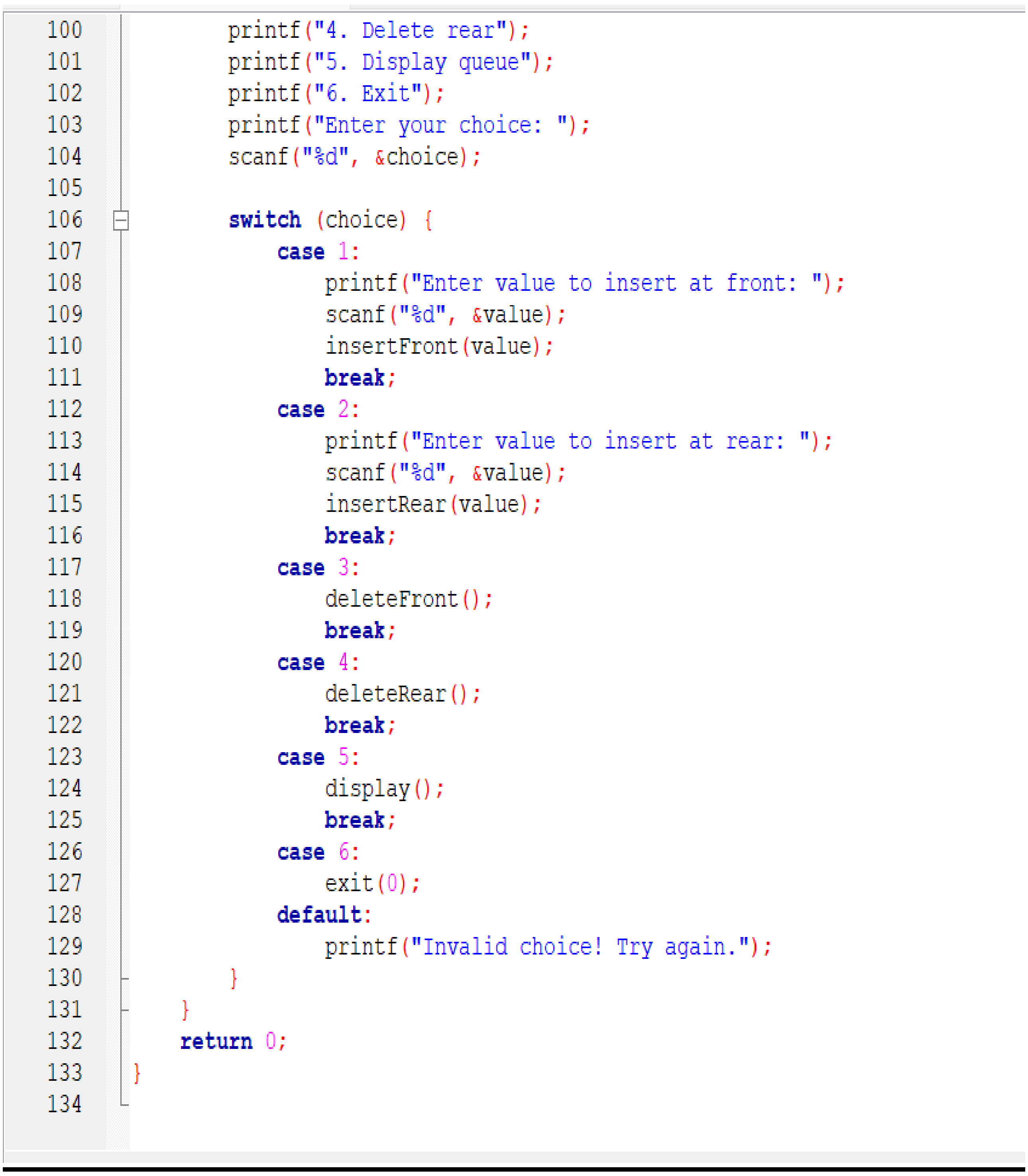
```c
#include <stdio.h>
#include <stdlib.h>

#define MAX 5

int dequeue[MAX];
int front = -1, rear = -1;

// Insert at the front end
void insertFront(int value) {
    if ((front == 0 && rear == MAX - 1) || (front == rear + 1)) {
        printf("Dequeue is full! Cannot insert.");
        return;
    }
    if (front == -1) { // First element insertion
        front = rear = 0;
    } else if (front == 0) {
        front = MAX - 1;
    } else {
        front -= 1;
    }
    dequeue[front] = value;
    printf("%d inserted at front.", value);
}

// Insert at the rear end
void insertRear(int value) {
    if ((front == 0 && rear == MAX - 1) || (front == rear + 1)) {
        printf("Dequeue is full! Cannot insert.");
        return;
    }
    if (front == -1) { // First element insertion
        front = rear = 0;
    } else if (rear == MAX - 1) {
        rear = 0;
    } else {
```



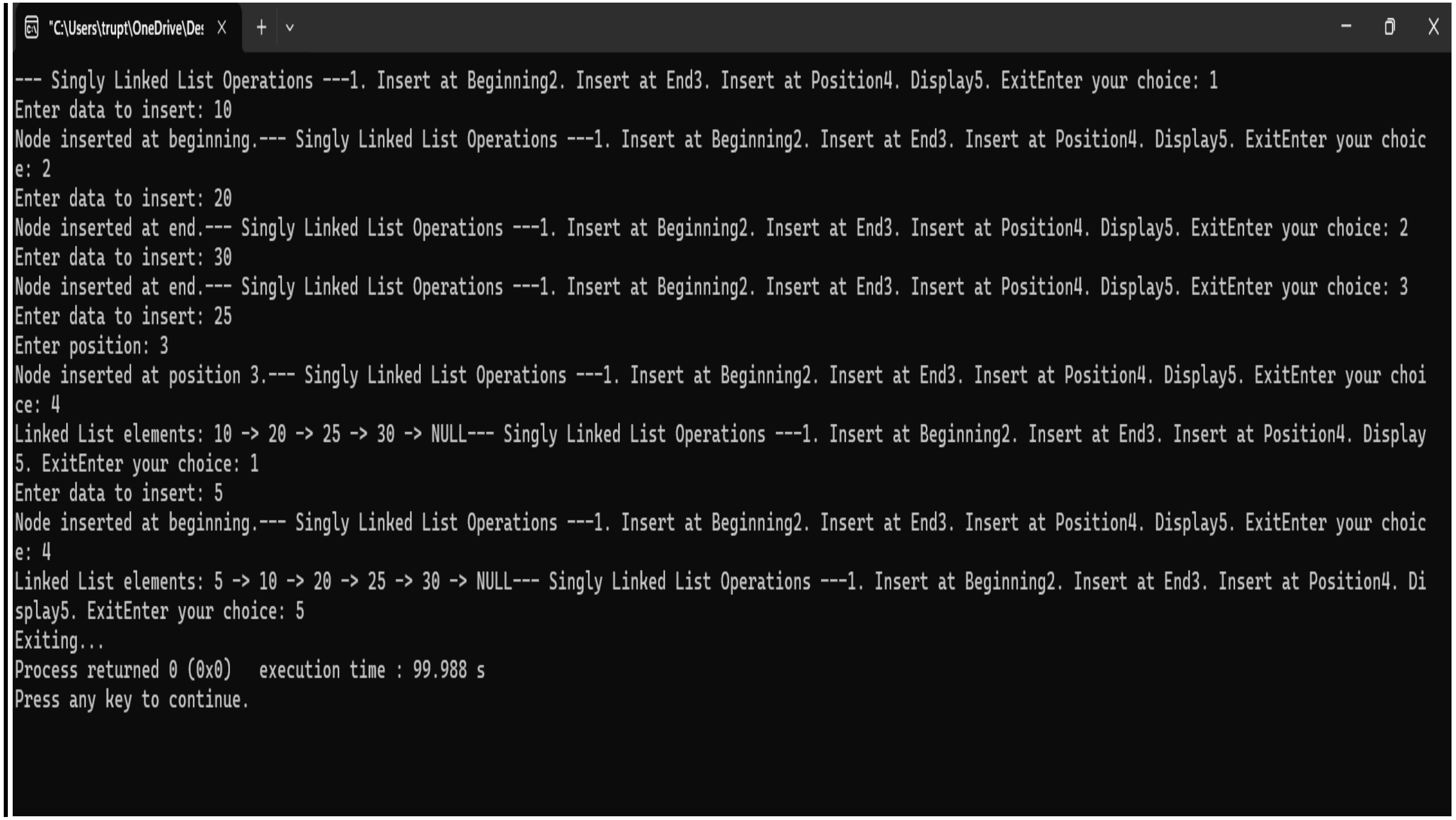
```c
    } else {
        rear += 1;
    }
    dequeue[rear] = value;
    printf("%d inserted at rear.", value);
}

// Delete from the front end
void deleteFront() {
    if (front == -1) {
        printf("Dequeue is empty! Cannot delete.");
        return;
    }
    printf("%d deleted from front.", dequeue[front]);
    if (front == rear) { // Only one element
        front = rear = -1;
    } else if (front == MAX - 1) {
        front = 0;
    } else {
        front++;
    }
}

// Delete from the rear end
void deleteRear() {
    if (front == -1) {
        printf("Dequeue is empty! Cannot delete.");
        return;
    }
    printf("%d deleted from rear.", dequeue[rear]);
    if (front == rear) { // Only one element
        front = rear = -1;
    } else if (rear == 0) {
        rear = MAX - 1;
    } else {
        rear--;
```



```c
        rear--;
    }
}

// Display the dequeue
void display() {
    if (front == -1) {
        printf("Dequeue is empty!");
        return;
    }
    printf("Dequeue elements are: ");
    int i = front;
    while (1) {
        printf("%d ", dequeue[i]);
        if (i == rear)
            break;
        i = (i + 1) %  MAX;
    }
    printf(" ");
}

int main() {
    int choice, value;

    while (1) {
        printf("--- DOUBLE ENDED QUEUE (DEQUE) ---");
        printf("1. Insert at front");
        printf("2. Insert at rear");
        printf("3. Delete front");
        printf("4. Delete rear");
        printf("5. Display queue");
        printf("6. Exit");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
```



```c
        printf("4. Delete rear");
        printf("5. Display queue");
        printf("6. Exit");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert at front: ");
                scanf("%d", &value);
                insertFront(value);
                break;
            case 2:
                printf("Enter value to insert at rear: ");
                scanf("%d", &value);
                insertRear(value);
                break;
            case 3:
                deleteFront();
                break;
            case 4:
                deleteRear();
                break;
            case 5:
                display();
                break;
            case 6:
                exit(0);
            default:
                printf("Invalid choice! Try again.");
        }
    }
    return 0;
}
```



```
--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 1
Enter data to insert: 10
Node inserted at beginning.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choic
e: 2
Enter data to insert: 20
Node inserted at end.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 2
Enter data to insert: 30
Node inserted at end.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 3
Enter data to insert: 25
Enter position: 3
Node inserted at position 3.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choi
ce: 4
Linked List elements: 10 -> 20 -> 25 -> 30 -> NULL--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display
5. ExitEnter your choice: 1
Enter data to insert: 5
Node inserted at beginning.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choic
e: 4
Linked List elements: 5 -> 10 -> 20 -> 25 -> 30 -> NULL--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Di
splay5. ExitEnter your choice: 5
Exiting...
Process returned 0 (0x0)   execution time : 99.988 s
Press any key to continue.
```