22_HEMANT_16.c

```c
/*
Roll no : 22
Batch: A
Author name: Hemant Gupta
Date: 23/08/2024
Description: Linked list implementation of queue
*/


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

// Define the structure for a queue node
struct Node {
    int data;
    struct Node* next;
};

// Define the structure for the queue
struct Queue {
    struct Node* front; // Pointer to the front node
    struct Node* rear;  // Pointer to the rear node
};

// Function to create a new queue
struct Queue* createQueue() {
    struct Queue* q = (struct Queue*)malloc(sizeof(struct Queue));
    q->front = q->rear = NULL; // Initialize both front and rear to NULL
    return q;
}

// Function to add an item to the queue
void enqueue(struct Queue* q, int value) {
    // Create a new node
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = value;
    new_node->next = NULL;

    // If the queue is empty, then new node is both front and rear
    if (q->rear == NULL) {
        q->front = q->rear = new_node;
        return;
    }

    // Add the new node at the end of the queue and change the rear
    q->rear->next = new_node;
    q->rear = new_node;
}

// Function to remove an item from the queue
int dequeue(struct Queue* q) {
```

```c
    // If the queue is empty, return -1 (or handle as needed)
    if (q->front == NULL) {
        printf("Queue is empty\n");
        return -1; // or some error value
    }

    // Store the front node and move the front pointer to the next node
    struct Node* temp = q->front;
    int value = temp->data;
    q->front = q->front->next;

    // If the front becomes NULL, then change rear also to NULL
    if (q->front == NULL)
        q->rear = NULL;

    free(temp); // Free the old front node
    return value;
}

// Function to display the queue
void display(struct Queue* q) {
    struct Node* temp = q->front;
    if (temp == NULL) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

// Main function to take user input and perform operations
int main() {
    struct Queue* q = createQueue();
    int choice, value;

    while (1) {
        printf("\nMenu:\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", &value);
                enqueue(q, value);
                break;
```

```
106
107             case 2:
108                 value = dequeue(q);
109                 if (value != -1)
110                     printf("Dequeued: %d\n", value);
111                 break;
112
113             case 3:
114                 display(q);
115                 break;
116
117             case 4:
118                 exit(0);
119
120             default:
121                 printf("Invalid choice\n");
122         }
123     }
124
125     return 0;
126 }
127
```