Dequeue operations using an array

Code:

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

#define MAX 5 // Maximum size of the deque

typedef struct {
   int data[MAX];
   int front;
   int rear;
} Deque;

// Function prototypes
void initializeDeque(Deque *dq);
int isFull(Deque *dq);
int isEmpty(Deque *dq);
void insertFront(Deque *dq, int value);
void insertRear(Deque *dq, int value);
int deleteFront(Deque *dq);
int deleteRear(Deque *dq);
void displayDeque(Deque *dq);

// Main function to test the deque
int main() {
   Deque dq;
   initializeDeque(&dq);

   int choice, value;

   while (1) {
      printf("\nDeque Operations:\n");
      printf("1. Insert Front\n");
      printf("2. Insert Rear\n");
      printf("3. Delete Front\n");
      printf("4. Delete Rear\n");
      printf("5. Display Deque\n");
      printf("6. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            printf("Enter the value to insert at the front: ");
            scanf("%d", &value);
            insertFront(&dq, value);
            break;
         case 2:
            printf("Enter the value to insert at the rear: ");
            scanf("%d", &value);
            insertRear(&dq, value);
            break;
         case 3:
            value = deleteFront(&dq);
            if (value != -1) {
               printf("Deleted value from front: %d\n", value);
            }
            break;
         case 4:
            value = deleteRear(&dq);
            if (value != -1) {
               printf("Deleted value from rear: %d\n", value);
            }
            break;
         case 5:
            displayDeque(&dq);
            break;
         case 6:
            exit(0);
```

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

    return 0;
}

// Initialize the deque
void initializeDeque(Deque *dq) {
    dq->front = -1;
    dq->rear = -1;
}

// Check if the deque is full
int isFull(Deque *dq) {
    return (dq->rear + 1) % MAX == dq->front;
}

// Check if the deque is empty
int isEmpty(Deque *dq) {
    return dq->front == -1;
}

// Insert an element at the front of the deque
void insertFront(Deque *dq, int value) {
    if (isFull(dq)) {
        printf("Deque is full! Cannot insert at the front.\n");
        return;
    }
    if (isEmpty(dq)) { // First element being inserted
        dq->front = dq->rear = 0;
    } else {
        dq->front = (dq->front - 1 + MAX) % MAX;
    }
    dq->data[dq->front] = value;
    printf("Inserted %d at the front.\n", value);
}

// Insert an element at the rear of the deque
void insertRear(Deque *dq, int value) {
    if (isFull(dq)) {
        printf("Deque is full! Cannot insert at the rear.\n");
        return;
    }
    if (isEmpty(dq)) { // First element being inserted
        dq->front = dq->rear = 0;
    } else {
        dq->rear = (dq->rear + 1) % MAX;
    }
    dq->data[dq->rear] = value;
    printf("Inserted %d at the rear.\n", value);
}

// Delete an element from the front of the deque
int deleteFront(Deque *dq) {
    if (isEmpty(dq)) {
        printf("Deque is empty! Cannot delete from the front.\n");
        return -1;
    }
    int value = dq->data[dq->front];
    if (dq->front == dq->rear) { // Only one element was present
        dq->front = dq->rear = -1;
    } else {
        dq->front = (dq->front + 1) % MAX;
    }
    return value;
}

// Delete an element from the rear of the deque
int deleteRear(Deque *dq) {
    if (isEmpty(dq)) {
        printf("Deque is empty! Cannot delete from the rear.\n");
```

```
    return -1;
  }
  int value = dq->data[dq->rear];
  if (dq->front == dq->rear) { // Only one element was present
    dq->front = dq->rear = -1;
  } else {
    dq->rear = (dq->rear - 1 + MAX) % MAX;
  }
  return value;
}

// Display all elements in the deque
void displayDeque(Deque *dq) {
  if (isEmpty(dq)) {
    printf("Deque is empty!\n");
    return;
  }
  printf("Deque elements: ");
  int i = dq->front;
  while (1) {
    printf("%d ", dq->data[i]);
    if (i == dq->rear) break;
    i = (i + 1) % MAX;
  }
  printf("\n");
}
```

OUTPUT:

```
Deque Operations:
1. Insert Front
2. Insert Rear
3. Delete Front
4. Delete Rear
5. Display Deque
6. Exit
Enter your choice: 1
Enter the value to insert at the front: 1
Inserted 1 at the front.

Deque Operations:
1. Insert Front
2. Insert Rear
3. Delete Front
4. Delete Rear
5. Display Deque
6. Exit
Enter your choice: 2
Enter the value to insert at the rear: 2
Inserted 2 at the rear.

Deque Operations:
1. Insert Front
2. Insert Rear
3. Delete Front
4. Delete Rear
5. Display Deque
6. Exit
Enter your choice: 5
Deque elements: 1 2

Deque Operations:
1. Insert Front
2. Insert Rear
3. Delete Front
4. Delete Rear
5. Display Deque
6. Exit
Enter your choice: 3
Deleted value from front: 1
```