P1.

```
#include <iostream>
using namespace std;

class Queue {
public:
    int* arr;
    int front;
    int rear;
    int capacity;

    Queue(int size) {
        capacity = size;
        arr = new int[capacity];
        front = -1;
        rear = -1;
    }

    bool isEmpty() {
        return front == -1;
    }

    bool isFull() {
        return rear == capacity - 1;
    }

    void enqueue(int value) {
        if (isFull()) {
            cout << "Queue Overflow: Unable to add " << value << " to the
queue!" << endl;
        } else {
            if (front == -1) {
                front = 0;
            }
            arr[++rear] = value;
            cout << value << " added to the queue." << endl;
        }
    }

    int dequeue() {
        if (isEmpty()) {
            cout << "Queue Underflow: Unable to remove from empty queue!"
<< endl;
            return -1;
        } else {
            int value = arr[front];
            if (front == rear) {
                front = rear = -1;
            } else {
                front++;
            }
            return value;
        }
    }
```

```
    int peek() {
        if (isEmpty()) {
            cout << "Queue is empty!" << endl;
            return -1;
        } else {
            return arr[front];
        }
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue is empty!" << endl;
        } else {
            cout << "Queue elements: ";
            for (int i = front; i <= rear; i++) {
                cout << arr[i] << " ";
            }
            cout << endl;
        }
    }
};

int main() {
    int size;
    cout << "Enter the size of the queue: ";
    cin >> size;

    Queue queue(size);

    int choice, value;

    do {
        cout << "\nQueue Operations:\n";
        cout << "1. Enqueue\n";
        cout << "2. Dequeue\n";
        cout << "3. Peek\n";
        cout << "4. Display\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;


        switch (choice) {
            case 1:
                cout << "Enter value to enqueue: ";
                cin >> value;
                queue.enqueue(value);
                break;

            case 2:
                value = queue.dequeue();
                if (value != -1) {
                    cout << value << " removed from the queue." << endl;
```

```
                }
                break;

            case 3:
                value = queue.peek();
                if (value != -1) {
                    cout << "Front element is: " << value << endl;
                }
                break;

            case 4:
                queue.display();
                break;

            case 5:
                cout << "Exiting..." << endl;
                break;

            default:
                cout << "Invalid choice! Please try again." << endl;
        }
    } while (choice != 5);

    return 0;
}
```

OUTPUT:

```
Enter the size of the queue: 4

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to enqueue: 10
10 added to the queue.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to enqueue: 16
16 added to the queue.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
```

```
Enter your choice: 1
Enter value to enqueue: 20
20 added to the queue.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to enqueue: 28
28 added to the queue.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 4
Queue elements: 10 16 20 28

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to enqueue: 13
Queue Overflow: Unable to add 13 to the queue!

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 3
Front element is: 10

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 2
10 removed from the queue.

Queue Operations:
1. Enqueue
2. Dequeue
```

```
3. Peek
4. Display
5. Exit
Enter your choice: 2
16 removed from the queue.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 4
Queue elements: 20 28

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 2
20 removed from the queue.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 3
Front element is: 28

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 2
28 removed from the queue.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 2
Queue Underflow: Unable to remove from empty queue!

Queue Operations:
1. Enqueue
2. Dequeue
```

```
3. Peek
4. Display
5. Exit
Enter your choice: 3
Queue is empty!

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 4
Queue is empty!

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 5
Exiting...
```

P2.

```
#include <iostream>
using namespace std;

class CircularQueue {
public:
    int* arr;
    int front;
    int rear;
    int capacity;

    CircularQueue(int size) {
        capacity = size;
        arr = new int[capacity];
        front = -1;
        rear = -1;
    }

    bool isEmpty() {
        return front == -1;
    }

    bool isFull() {
        return (rear + 1) % capacity == front;
    }

    void enqueue(int value) {
        if (isFull()) {
```

```
            cout << "Queue Overflow: Unable to add " << value << " to the
queue!" << endl;
        } else {
            if (front == -1) {
                front = 0;
            }
            rear = (rear + 1) % capacity;
            arr[rear] = value;
            cout << value << " added to the queue." << endl;
        }
    }

    int dequeue() {
        if (isEmpty()) {
            cout << "Queue Underflow: Unable to remove from empty queue!"
<< endl;
            return -1;
        } else {
            int value = arr[front];
            if (front == rear) {
                front = rear = -1; // Queue becomes empty
            } else {
                front = (front + 1) % capacity;
            }
            return value;
        }
    }

    int peek() {
        if (isEmpty()) {
            cout << "Queue is empty!" << endl;
            return -1;
        } else {
            return arr[front];
        }
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue is empty!" << endl;
        } else {
            cout << "Queue elements: ";
            int i = front;
            while (i != rear) {
                cout << arr[i] << " ";
                i = (i + 1) % capacity;
            }
            cout << arr[rear] << endl;
        }
    }
};

int main() {
    int size;
```

```
    cout << "Enter the size of the circular queue: ";
    cin >> size;

    CircularQueue queue(size);

    int choice, value;

    do {
        cout << "\nCircular Queue Operations:\n";
        cout << "1. Enqueue\n";
        cout << "2. Dequeue\n";
        cout << "3. Peek\n";
        cout << "4. Display\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value to enqueue: ";
                cin >> value;
                queue.enqueue(value);
                break;

            case 2:
                value = queue.dequeue();
                if (value != -1) {
                    cout << value << " removed from the queue." << endl;
                }
                break;

            case 3:
                value = queue.peek();
                if (value != -1) {
                    cout << "Front element is: " << value << endl;
                }
                break;

            case 4:
                queue.display();
                break;

            case 5:
                cout << "Exiting..." << endl;
                break;

            default:
                cout << "Invalid choice! Please try again." << endl;
        }
    } while (choice != 5);

    return 0;
}
```

```
OUTPUT:
Enter the size of the circular queue: 4

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to enqueue: 10
10 added to the queue.

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to enqueue: 16
16 added to the queue.

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to enqueue: 20
20 added to the queue.

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to enqueue: 28
28 added to the queue.

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 4
Queue elements: 10 16 20 28

Circular Queue Operations:
```

```
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 3
Front element is: 10

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to enqueue: 13
Queue Overflow: Unable to add 13 to the queue!

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 2
10 removed from the queue.


Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 2
16 removed from the queue.

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 3
Front element is: 20

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 4
Queue elements: 20 28
```

```
Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 2
20 removed from the queue.

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 2
28 removed from the queue.

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 3
Queue is empty!

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 4
Queue is empty!

Circular Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter your choice: 5
Exiting...
```