```
public class DequeUsingQueue {
    private Queue<T> q;
    private int n;

    public DequeUsingQueue() {
        q = new Queue<T>();
        n = 0;
    }

    public void addFirst(T e) {
        q.enqueue(e);
        for(int i = 0; i < q.length() - 1; i++)
            q.enqueue(q.serve());
        n++;
    }

    public void addLast(T e) {
        q.enqueue(e);
        n++;
    }

    public T removeFirst() {
        n--;
        return q.serve();
    }

    public T getFirst() {
        T x = q.serve();
        q.enqueue(x);
        for(int i = 0; i < q.length() - 1; i++)
            q.enqueue(q.serve());
        return x;
    }

    public T getLast() {
        for(int i = 0; i < q.length() - 1; i++)
            q.enqueue(q.serve());
        T x = q.serve();
        q.enqueue(x);
        return x;
    }

    public int size() {
        return n;
    }

    public boolean isEmpty() {
        return n == 0;
    }
}

public static<T> void removeEquals(Queue<T> q, T e) {
    int n = q.length();
    for(int i = 0; i < n; i++) {
        T x = q.serve();
        if(!x.equals(e))
            q.enqueue(x);
    }
}

public static<T> boolean find(Queue<T> q, T e) {
    boolean found = false;
    int n = q.length();
    for(int i = 0; i < n; i++) {
        T x = q.serve();
        if(x.equals(e))
            found = true;
        q.enqueue(x);
    }

    return found;
}
```