Trường Đại Học Quốc Tế - ĐHQG TP.HCM

LAB REPORT

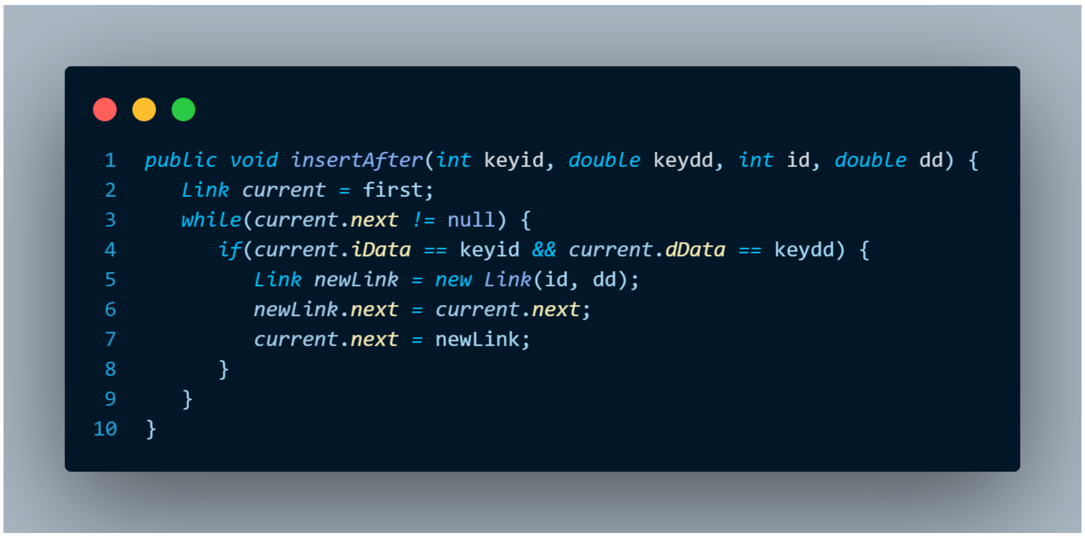
Course: Algorithms & Data Structures LAB 4

Full Name: Trần Minh Phúc . . . .

Student's ID: ITCSIU24070 . . . .

LinkList2App.java

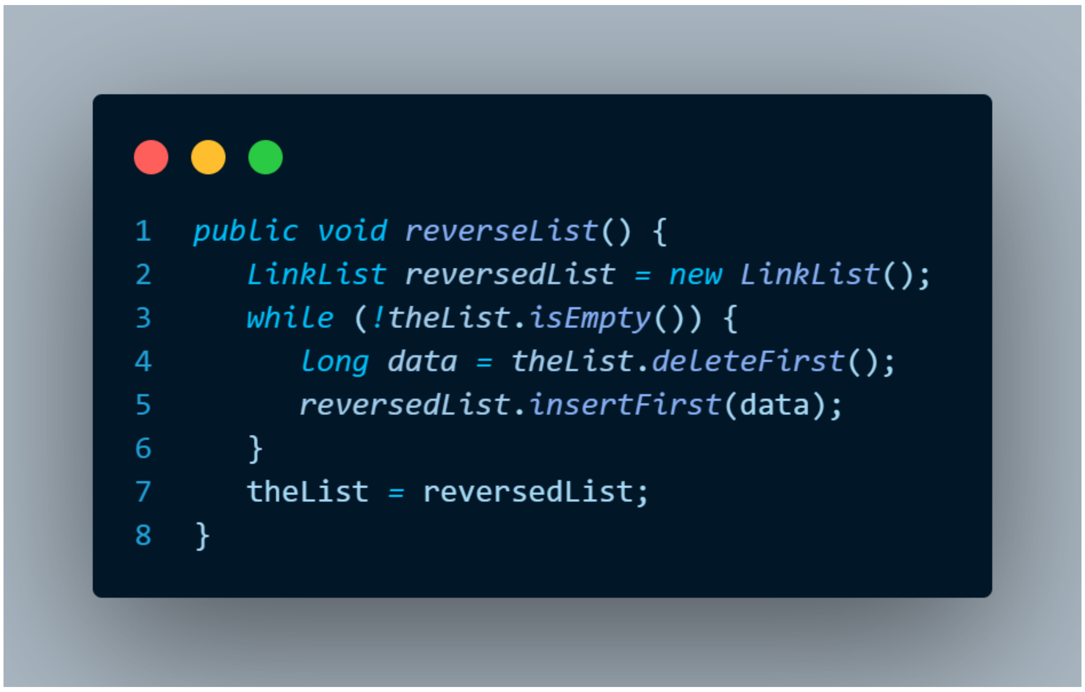
Add a method insertAfter to insert after a particular item in this list



```java
public void insertAfter(int keyid, double keydd, int id, double dd) {
    Link current = first;
    while(current.next != null) {
        if(current.iData == keyid && current.dData == keydd) {
            Link newLink = new Link(id, dd);
            newLink.next = current.next;
            current.next = newLink;
        }
    }
}
```

LinkStackApp.java

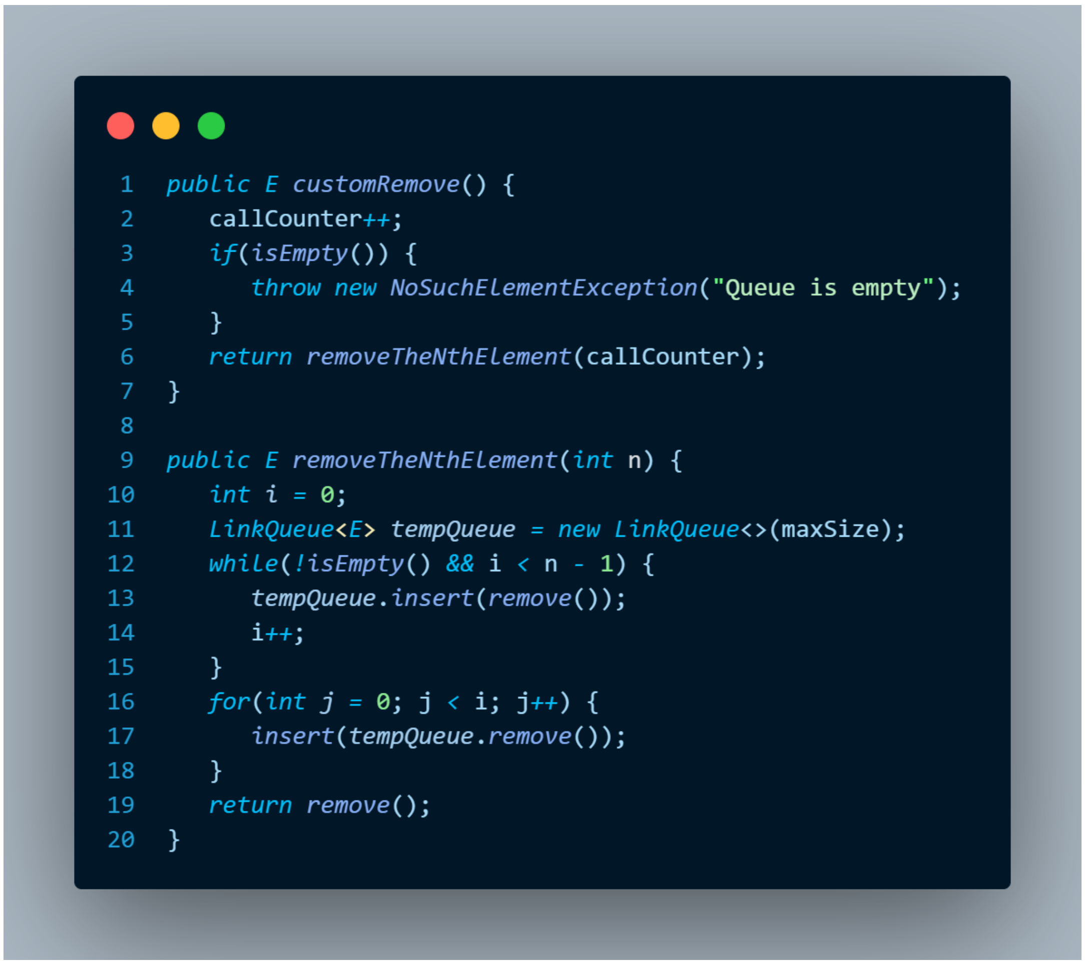
Write an application to reverse a list using a stack



```java
public void reverseList() {
    LinkList reversedList = new LinkList();
    while (!theList.isEmpty()) {
        long data = theList.deleteFirst();
        reversedList.insertFirst(data);
    }
    theList = reversedList;
}
```

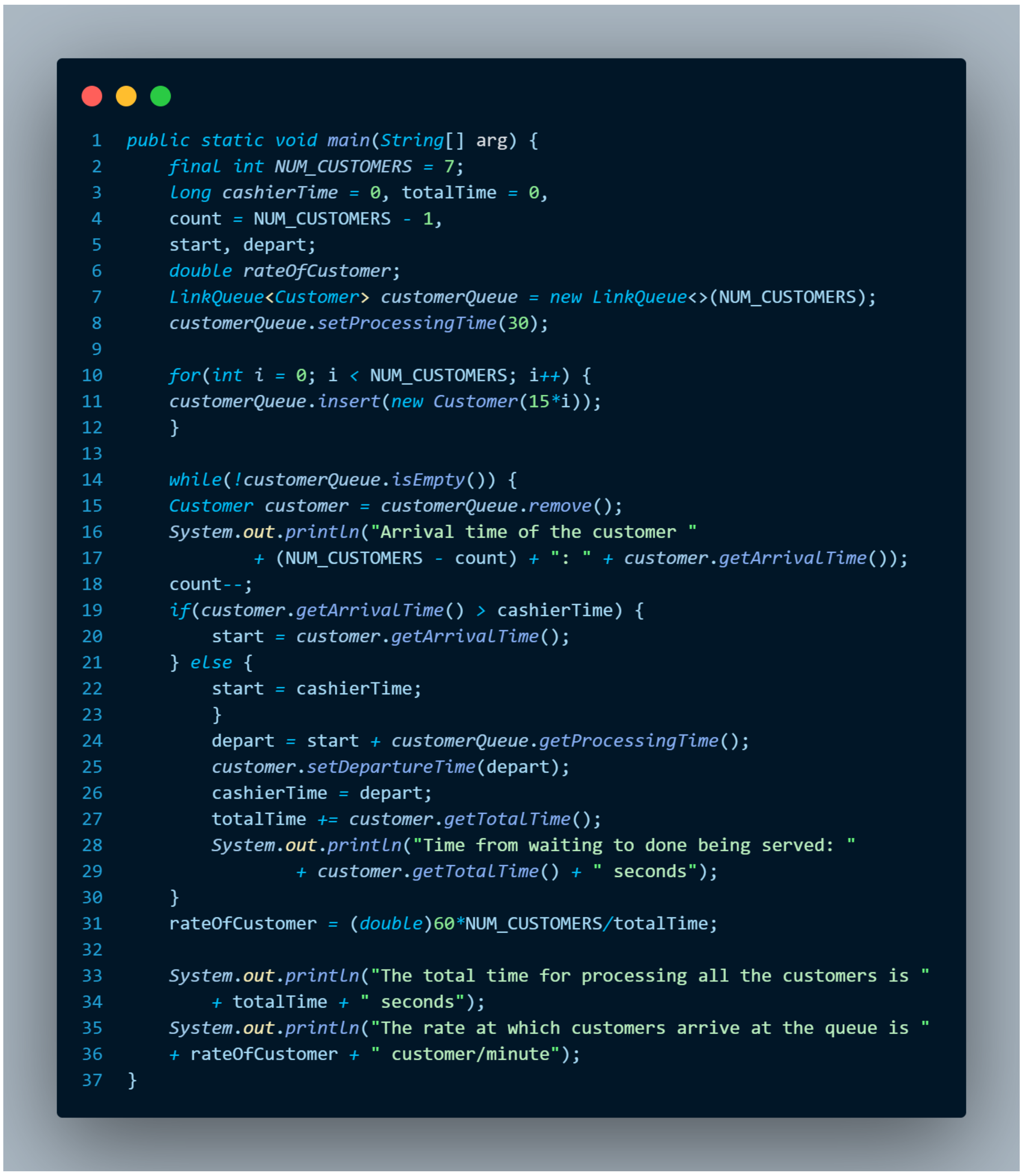
LinkQueueApp.java

- Create a new remove() method that removes item N after N calls to the method.

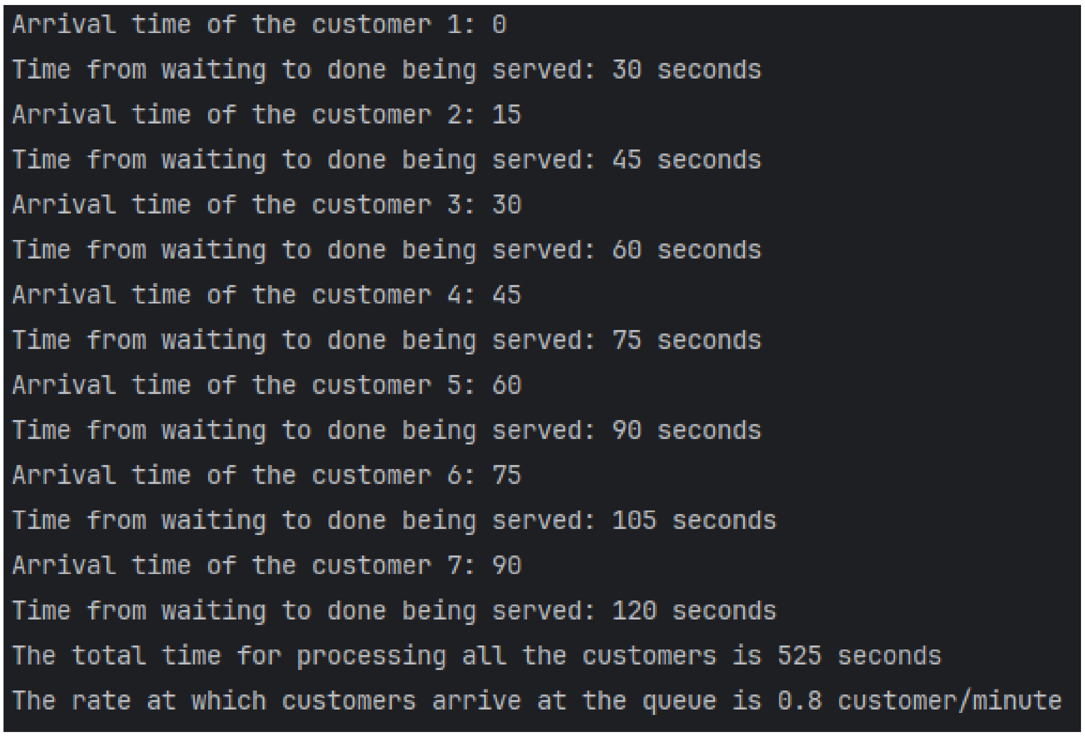


```java
public E customRemove() {
    callCounter++;
    if(isEmpty()) {
        throw new NoSuchElementException("Queue is empty");
    }
    return removeTheNthElement(callCounter);
}

public E removeTheNthElement(int n) {
    int i = 0;
    LinkQueue<E> tempQueue = new LinkQueue<>(maxSize);
    while(!isEmpty() && i < n - 1) {
        tempQueue.insert(remove());
        i++;
    }
    for(int j = 0; j < i; j++) {
        insert(tempQueue.remove());
    }
    return remove();
}
```

- Simulate a queue of customers each one served for a random amount of time.

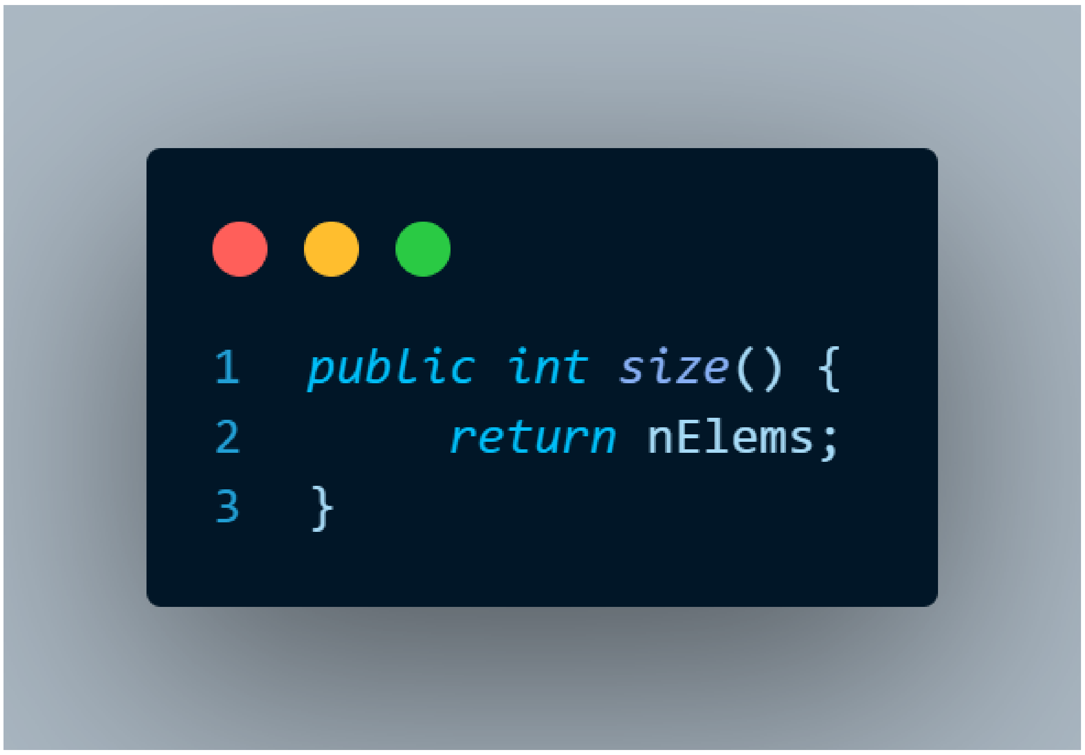


```java
public static void main(String[] arg) {
    final int NUM_CUSTOMERS = 7;
    long cashierTime = 0, totalTime = 0,
    count = NUM_CUSTOMERS - 1,
    start, depart;
    double rateOfCustomer;
    LinkQueue<Customer> customerQueue = new LinkQueue<>(NUM_CUSTOMERS);
    customerQueue.setProcessingTime(30);

    for(int i = 0; i < NUM_CUSTOMERS; i++) {
    customerQueue.insert(new Customer(15*i));
    }

    while(!customerQueue.isEmpty()) {
    Customer customer = customerQueue.remove();
    System.out.println("Arrival time of the customer "
            + (NUM_CUSTOMERS - count) + ": " + customer.getArrivalTime());
    count--;
    if(customer.getArrivalTime() > cashierTime) {
        start = customer.getArrivalTime();
    } else {
        start = cashierTime;
        }
        depart = start + customerQueue.getProcessingTime();
        customer.setDepartureTime(depart);
        cashierTime = depart;
        totalTime += customer.getTotalTime();
        System.out.println("Time from waiting to done being served: "
                + customer.getTotalTime() + " seconds");
    }
    rateOfCustomer = (double)60*NUM_CUSTOMERS/totalTime;

    System.out.println("The total time for processing all the customers is "
        + totalTime + " seconds");
    System.out.println("The rate at which customers arrive at the queue is "
    + rateOfCustomer + " customer/minute");
}
```



```
Arrival time of the customer 1: 0
Time from waiting to done being served: 30 seconds
Arrival time of the customer 2: 15
Time from waiting to done being served: 45 seconds
Arrival time of the customer 3: 30
Time from waiting to done being served: 60 seconds
Arrival time of the customer 4: 45
Time from waiting to done being served: 75 seconds
Arrival time of the customer 5: 60
Time from waiting to done being served: 90 seconds
Arrival time of the customer 6: 75
Time from waiting to done being served: 105 seconds
Arrival time of the customer 7: 90
Time from waiting to done being served: 120 seconds
The total time for processing all the customers is 525 seconds
The rate at which customers arrive at the queue is 0.8 customer/minute
```

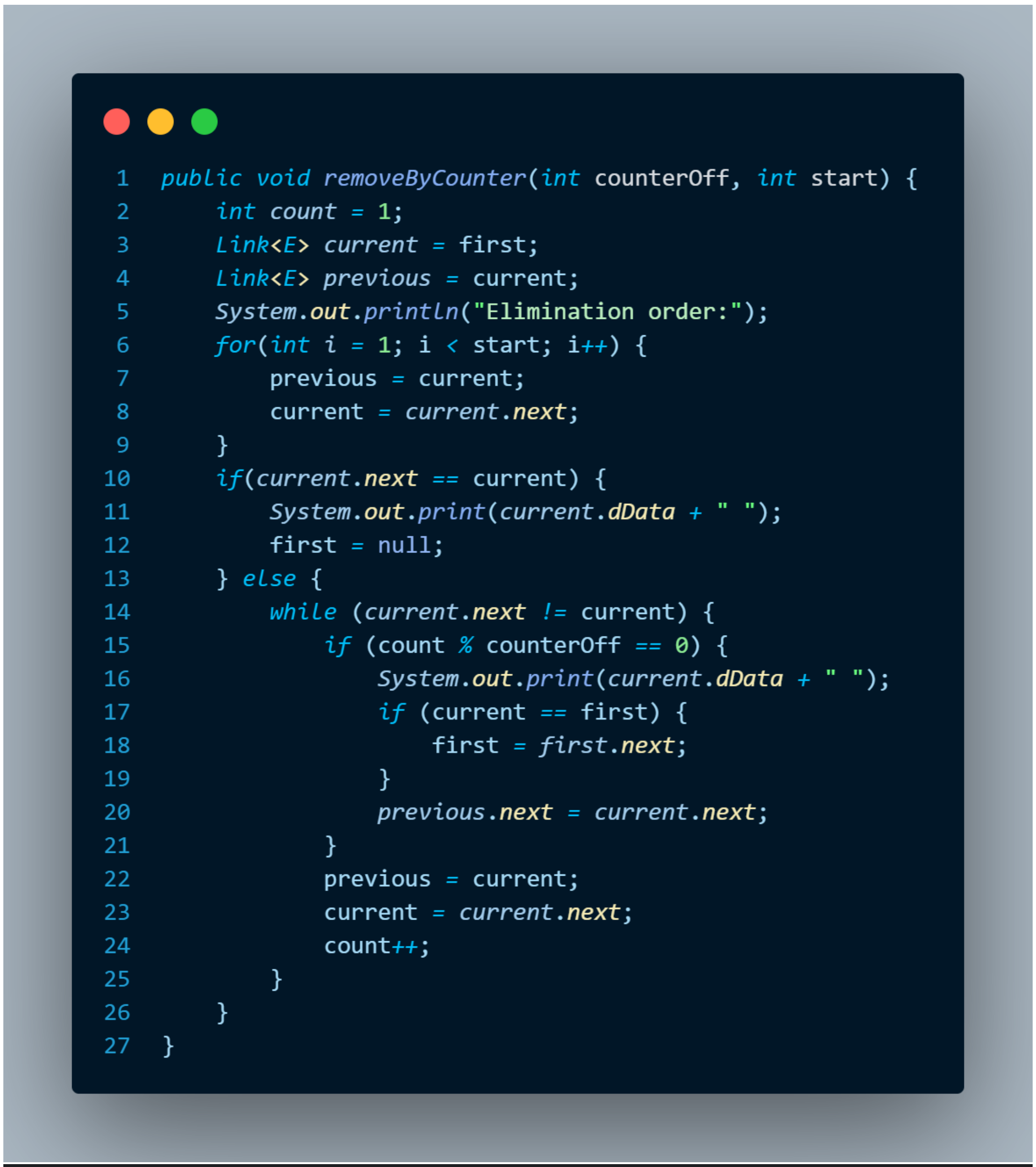
- **Add a size() method and investigate how simulation is affected by the time needed to serve a customer and the rate at which customers join the queue.**



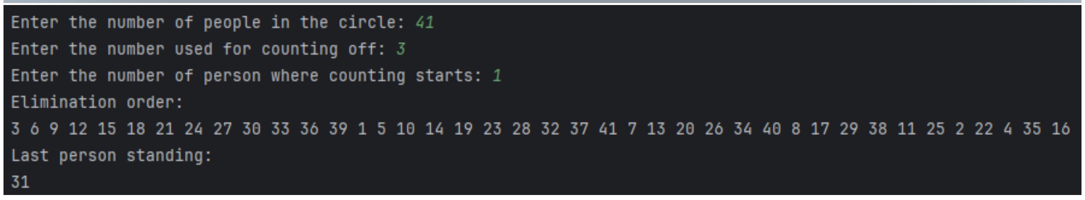
```java
public int size() {
        return nElems;
}
```

If the range of time processing is more narrowed than the arrival time, the waiting time processing time only takes processing time into account, meaning the customers do not have to wait for the queue.

Josephus Problem



```java
public void removeByCounter(int counterOff, int start) {
    int count = 1;
    Link<E> current = first;
    Link<E> previous = current;
    System.out.println("Elimination order:");
    for(int i = 1; i < start; i++) {
        previous = current;
        current = current.next;
    }
    if(current.next == current) {
        System.out.print(current.dData + " ");
        first = null;
    } else {
        while (current.next != current) {
            if (count % counterOff == 0) {
                System.out.print(current.dData + " ");
                if (current == first) {
                    first = first.next;
                }
                previous.next = current.next;
            }
            previous = current;
            current = current.next;
            count++;
        }
    }
}
```



```
Enter the number of people in the circle: 41
Enter the number used for counting off: 3
Enter the number of person where counting starts: 1
Elimination order:
3 6 9 12 15 18 21 24 27 30 33 36 39 1 5 10 14 19 23 28 32 37 41 7 13 20 26 34 40 8 17 29 38 11 25 2 22 4 35 16
Last person standing:
31
```

This is the end of the report