Overview
For this week's session plan, we have decided that we should conduct a review session for the next upcoming exam this Thursday and Friday. In order to achieve a good learning session, we agreed on addressing certain questions students may have from previous session, and then providing with our own set of review problems to the class (all problems may be answered in a cooperative learning manner as always). The main topics to review are queues, stacks, and link lists. Since recursion also plays an important part in some operation for this data structures, we will also apply them as a previous review for the class. At the end of the session, students should be ready for the upcoming exam.

Lesson Plan:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time(mins)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questions. Anyone?</td>
<td>5</td>
</tr>
<tr>
<td>group into groups of 2 or 3</td>
<td>2</td>
</tr>
<tr>
<td>Go over basic structure of Link List</td>
<td>5</td>
</tr>
<tr>
<td>Review stack characteristics</td>
<td>5</td>
</tr>
<tr>
<td>Definition and implementation of queues and circular queue</td>
<td>8</td>
</tr>
<tr>
<td>Do exercises</td>
<td>15</td>
</tr>
<tr>
<td>Go over exercises as a group</td>
<td>10</td>
</tr>
</tbody>
</table>

Total Time = 50 minutes.

Note. -This plan can vary depending on the participation of the group and the number of students in the session.

Before starting with the exercises, we believe it is convenient to review all data structure first and then let them work on the exercises as a group and answering the questions as a class. The exercises will be the following:

**Link List**

```java
//Gives the sum of a list recursively
public static int sum (LN l)
{
    if (l == null)
        return 0;
```
else
    return l.value + sum(l.next);
}

// Searches for a value in the list recursively
public static LN search (LN l, int value)
{
    if (l == null)
        return null;
    else
        if (l.value == value)
            return l;
        else
            return search(l.next, value);
}

Note: For the 2 above, please just insert the comments to your copies for the students. I just provided a sample answer for simplicity.

// Trace the following operations on a link list. Include the state of the list at every operation and // the value of the head.

LinkedList list = new LinkedList();
LinkedList list2 = new LinkedList();
insert(list, Node a);
insert(list, Node b);
insert(list, Node c);
remove(list, Node b);
find(list2Node c);
concatinate(list, list2);

Stacks
// Give pseudo-code for any useful implementation of a stack not seen in class.

Queues
// What is the difference between a queue and a circular queue?

// Show the state of the queue at every operation. For each, show the top and bottom pointing value of the queue.

Queue q = new Queue(4);
enqueue(1);
enqueue(2);
enqueue(3);
dequeue();
We decided in these types of exercises because of 2 main reasons: Some students have problems understanding the methods of a data structure and because tracing is a must in an exam. Also, students need help with the code in a method, we can also provide in explain the reason behind it with no problem. If there are more questions regarding the exam or the class, students are invited to make appointment with us as always.