Topic: Exam Review

Activity Guidelines

Group Size: 3

Method of Assigning Students: Count the number of students in the class, divide by 3, count off from 1 to the quotient, and group identical numbers.

Materials:

✓ Handout (one copy per group) with questions to be answered at the end of the session

Roles:

Coordinator/Leader: Clarifies goals and objectives, allocates roles for each team member and divides the tasks within the group.

Monitor/Evaluator: Person designed to evaluate the different ideas to approach the problem and make an accurate judgment of the most beneficial option.

Implementer: Person in charge to transform discussions and ideas into a technical solution for the given problem.

Individual Accountability: Each team member gets assigned a specific role in order to ensure every student within a team participates and contributes to reach a solution for each problem presented in the activity.

Activity Summary

Every team is expected to implement different methods that require basic knowledge on both linked list and recursion.

Section 1: This section requires for students to develop a solution for 4 problems that tests their knowledge on linked list. Additionally, they must be able to traduce an iterative method into a recursive method, and vice versa.

Section 2: This section is based on the most frequently ask questions about recursion and linked list. Students must have a complete understanding in both topics in order to succeed.
Elementary Data Structures
Peer Session

Note: Some of these questions were taken from a previous Elementary Data Structures Midterm Exam.

Section 1 - Linked List

1. Suppose you have a linked list of Node objects with double data elements, as defined below. Write a Java method that returns a count of the number of values in the list between a given minimum value and maximum value (e.g., a count of the numbers between 1.5 and 6.3). Write your code in the space provided below.

   Answer:
   ```java
   int countValues(Node head, double minValue, double maxValue) {
       int count = 0;
       if(head!=null) {
           while(head.next!=null) {
               if(head.data > minValue && head.data < maxValue)
                   count++;
               head = head.next;
           }
       }
       return count;
   }
   ```

2. Suppose you have a singly-linked list of Node objects with double data elements. Write a Java method that will return the head reference to a new linked list with the same elements, but in reversed order. The new list can use the same node objects as the original. Hint: think about reversing the links in the list; try writing down an example to see how the links need to change.

   Answer:
   ```java
   public Node reverse(Node head) {
       Node before = null;
       Node tmp = head;
       while(tmp!=null) {
           Node next = tmp.next;
           tmp.next = before;
           before = tmp;
           tmp = next;
       }
       head = before;
       return head;
   }
   ```
3. Write an iterative method for doing a sequential search in a linked list with String data elements. The method should take in a String to search for in the list, and a reference to the head Node in the list. The method should return a reference to the Node containing the required data if it is found, and null if the String is not in the list. You may assume that you have a standard Node class already implemented.

**Answer:**

```java
public Node isFound(Node head, String str) {
    if(head!=null) {
        while(head.next!=null) {
            if((head.data).equals(str))
                return head;
            head = head.next;
        }
    }
    return null;
}
```

4. Write a recursive method for doing a sequential search in a linked list with String data elements. The method should take in a String to search for in the list, and a reference to the head Node in the list. The method should return a reference to the Node containing the required data if it is found and null if the String is not in the list. You may assume you have a standard Node class already implemented.

**Answer:**

```java
public Node isFound(Node head, String str) {
    if(head==null) {
        return null;
    } else if(((head.data).equals(str))
        return head;
    else
        return isFound(head.next,str);
}
```
SECTION 2 - QUESTIONS

For each of the questions below, select the answers(s) to address the question. Write the letter(s) of your answer on the provided line. Some of the questions might have multiple answers.

(a) What are the base cases in the following recursive method?

```java
public static void xMethod(int n) {
    if(n > 0) {
        System.out.println(n%10);
        xMethod(n/10);
    }
}
```

i. n<=0
ii. n<0
iii. no base cases
iv. n>0

Answer: n<= 0

(b) What will be displayed by invoking the xfunction (6)?

```java
public static int xfunction(int n) {
    if(n<=1) return 1;
    return n + xfunction(n-2);
}
```

i. 14
ii. 13
iii. 12
iv. 11

Answer: 12

(c) A linked list is a data structure consisting of a group of nodes which together represent a sequence just like an array. One can directly access any element of a random location of the linked list just by using the index of the element or the “.” Operator.

A. The statement is true.
B. The statement is false

Answer: The statement is false.
(d) Consider the following code that is supposed to recursively count how many ways a particular spaghetti bran can break into two.

```java
public static int split(int len) {
    // Split even spaghetti into two pieces:
    if (len % 2 == 0)
        return split(len/2) + split(len/2);
    return 1 + split(len-1);
}
```

i. What is the most significant error in the code above?

**Answer:** No base case is set when `len` is equal or less than 0.

ii. When would you expect to see a problem with the code above – When you compile and/or execute in a Java Virtual Machine?

  Compile Problem? Yes / No
  Execution Problem? Yes / No