Data Structures Midterm

Your Name______________________________________________________

Teaching Fellow (circle one) Hans Jennifer Maureen

Answer all questions in the test itself. You may use the backs of pages. If any question appears ambiguous or peculiar, ask for help! It would be a shame to misinterpret a question which you are capable of answering. If you need to make assumptions, jot them down with your answer. You have two hours to complete the exam.

This is an “open book” exam, so you may use any notes, texts or other references at your disposal. However, do not waste too much time leafing through your materials. The point value of each problem should give you a rough idea of how to budget your time. (plan on spending slightly more than 1 minute per point).

**Part A** 10 Multiple-Choice Questions (50 points)
(Each question in part A is worth 5 points. For each, write your answer in the space provided below)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

For each question in Section A, select only one response. Pick the response that comes closest to answering the problem.
[1] In a single function, what is the maximum number of statements that may be recursive calls?
A. 1
B. 2
C. N (where N is the argument)
D. There is no fixed maximum

[2] Consider the following function.

```java
static void test(int a, int b)
{
    if (a < b)
        return 0;
    else
        return (1 + f(a-b, b));
}
```

What is printed by the call test(15, 4)?
A. 1
B. 2
C. 3
D. 4
E. 5

[3] What additional requirement is placed on an array, so that binary search may be used to locate an entry?
A. The array elements must have an odd number of entries.
B. The array must have at least 2 entries.
C. The array must be sorted.
D. The array’s size must be a power of two.
E. None of the above are required

[4] What is the post-fix representation of the expression below?

```
1 + 2 * 3 + 4 * 5
```

A. 12+3*45**
B. 123++45**
C. 123*+45**
D. 12+3*45**
E. 123*4+5**
F. none of the above
[5] If the binary tree

![Binary Tree Diagram]

were listed via *preorder* traversal, the result would be

A. 6 17 22 9 4 16 12  
B. 9 4 17 16 12 22 6  
C. 9 17 6 4 16 22 12  
D. 6 17 9 4 22 16 12  
E. 6 9 17 4 16 22 12

[6] The *binary search tree* shown below was constructed by inserting items as they arrived. It could be constructed for all but one of the following input sequences - which one will **not** produce this search tree?

![Binary Search Tree Diagram]

A. 5 9 3 7 6 8 4 12 20  
B. 5 9 3 6 7 8 4 12 20  
C. 5 3 4 9 12 7 8 6 20  
D. 5 9 7 8 6 12 20 3 4  
E. 5 9 7 3 8 12 6 4 20
Questions [7] and [8] both refer to the following diagram:

```
head
    2
    ▼
    ▼
    ▼
    9
    ▼
    ▼
    ▼
    7
    ▼
    ▼
    ▼
null
```

The diagram represents a linked-list of integers that can be built using the declarations.

```
class Node {
    int value;
    Node next;
}
Node head;
```

[7] When applied to the original list illustrated above, what will be the effect of the code below:

```
Node temp = head;

while (temp.next != null)
    temp = temp.next;

System.out.println(temp.value);
```

A. It will print 2.
B. It will print 9.
C. It will print 7.
D. It will cause a compile-time error.
E. When applied to the list above, it will cause a run-time error.

[8] What will be the effect of the statement below?

```
head.next.value = head.next.next.value;
```

A. It will convert the above list from "2 9 7" to "2 7".
B. It will convert the above list from "2 9 7" to "2 7 7".
C. It will convert the above list from "2 9 7" to "2 9 7 7".
D. It will convert the above list from "2 9 7" to "2 9".
E. It will cause a compile-time error.
F. When applied to the list above, it will cause a run-time error.
Questions [9] and [10] both refer to the following function

```java
void super_write_vertical(int number) {
    if (number < 0) {
        System.out.println('-');
        super_write_vertical(-1* number);
    } else if (number < 10)
        System.out.println(number);
    else {
        super_write_vertical(number/10);
        System.out.println(number % 10);
    }
}
```

[9] What values of `number` are handled by the function without a recursive call?
A. number < 0  
B. number < 10  
C. ((number >= 0) && (number < 10))  
D. number > 10

[10] Which of the calls below will result in the most recursive calls?
A. super_write_vertical(1000)  
B. super_write_vertical(159)  
C. super_write_vertical(0)  
D. super_write_vertical(1026)  
E. super_write_vertical(-1023)
Part B -- Short Answer Problems (25 points total)
Write on the backs of pages in the test

[11] (10 points) Tokens  
A token in a certain programming language begins with a letter, and continues with letters or digits. Thus “head” and “tail15” are tokens, but “back-bend” is not. You are to devise a way to recognize tokens.

[A] (5 points)  Sketch a finite state machine that recognizes Tokens
[B] (5 points)  Write a program that takes an array of char and looks for the first token. You may use these functions defined in class Character.

    public static boolean isLetter(char ch);
    public static boolean isDigit(char ch);

[12] 15 points  Assume that the definitions below are used to hold a sorted linked list, with items in ascending order. Write a function or method that removes items in the list that occur twice. Since the list is sorted, duplicates will be adjacent. You may decide what you do with items that appear more than once, but be sure to describe what your function does.

    class intNode {
        int value;
        intNode next;
    }
    intNode head;

    Before Removing Duplicates

    After Removing Duplicates

    [12a] (3 points)  Write a prototype (declare your function)
    [12b] (12 points)  Write your function
Part C - Main Programming Problem

[13] (25 points) The following definition of Node is used to build strings. Write a function that will take a text and a pattern, and search for the pattern as a sub-string of the text. Your function should return the index of the sub-string in the text, or -1 if the pattern is not found.

```java
class Node {
    char value;
    Node next;
}
Node text, pat, mike;

// Function prototype
static int match(Node text, Node pattern);
```

Using the sample strings illustrated above,

```java
match(text, pat);
would return 3  (a match is found in the third position)
```

```java
match(text, mike);
would return -1  (the pattern is not found in the text).
```