Points off | 1 | 2 | 3 | 4 | 5 | 6 | Total off | Net Score
---|---|---|---|---|---|---|---|---

CS 314 – Exam 2 – Spring 2019

Your Name: ____________________________________________

Your UTEID: ____________________________________________

Circle your TAs Name: Aish Amir Anthony Ethan Fatima
                      Hailey Jacob Lucas Rebecca Terrell

Instructions:
1. There are 6 questions on this test. 100 points available. Scores will be scaled to 200 points.
2. You have 2 hours to complete the test.
3. Place your final answers on this test. Not on the scratch paper. Answer in pencil.
4. You may not receive any outside assistance. No electronic devices or calculators may be used.
5. When answering coding questions, ensure you follow the restrictions of the question.
6. Do not write code to check the preconditions.
7. On coding questions, you may implement your own helper methods unless disallowed by the question.
8. On coding questions make your solutions as efficient as possible given the restrictions of the question.
9. Test proctors will not answer any questions regarding the content of the exam. If you think a question is ambiguous or has an error, state your assumptions and answer based on those assumptions.
10. When you complete the test show the proctor your UTID, give them the test and all the scratch paper, used or not, and leave the room quietly.

1. (1 point each, 15 points total) Short answer. Place your answer on the line next to or under the question. Assume all necessary imports have been made.
   a. If a question contains a syntax error or compile error, answer compile error.
   b. If a question would result in a runtime error or exception, answer runtime error.
   c. If a question results in an infinite loop, answer infinite loop.
   d. Assume \( \log_2(1,000) = 10 \) and \( \log_2(1,000,000) = 20 \).

A. What is returned by the method call \( a(1) \)?

```java
public static int a(int n) {
    if (n <= -3)
        return 5;
    else
        return a(n - 1) + Math.abs(n) * 2;
}
```
B. What is output by the following code?

```java
System.out.print(b(2, 10));

public static int b(int x, int y) {
    if (x <= y) {
        int t = x - y;
        t += b(x + 2, y - 1);
        return t;
    } else {
        return 10;
    }
}
```

C. What is output by the following code?

```java
c(1, "ALGO");

public static void c(int i, String s) {
    if (i >= s.length())
        System.out.print("*");
    else {
        if (i % 2 == 0) {
            System.out.print(i);
            c(i + 1, s);
            System.out.print(s.charAt(i));
        } else {
            System.out.print(s.charAt(i));
            c(i + 1, s);
            System.out.print(i);
        }
    }
}
```

D. What is output by the following code?

```java
System.out.print(d(10));

public static int d(int n) {
    if (n <= 7)
        return 10;
    else
        return d(n - 2) + 2 + d(n - 1);
}
E. You are going to implement a Queue and use a circular, doubly linked list as the internal storage container. Which end of the list (beginning or end) is best to use as the front of the Queue? Why?

___________________________________________________________________

F. A method uses the insertion sort algorithm presented in class to sort data into descending order. It takes the method 1 second to complete given an array of 10,000 distinct elements sorted in ascending order. How long do you expect the method to complete given an array of 30,000 distinct elements sorted in ascending order?

______________________________________________________________

G. In lecture we discussed five sorting algorithms: selection sort, insertion sort, radix sort, quicksort, and mergesort. Of these 5 sorts, which ones are typically in-place algorithms?

______________________________________________________________

H. The following method takes 0.05 seconds to complete when list.size() = 100,000. What is the expected time for the method to complete when list.size() = 400,000? Assume tgt occurs once in the list and assume tgt is typically halfway through the list. Assume the lengths of the Strings in list are always between 2 and 10 inclusive.

_________________________________________________________________

// pre: tgt != null
public static int methodH(LinkedList<String> list, String tgt) {
    int index = -1;
    for (String s : list) {
        index++;
        if (tgt.equals(s)) {
            list.remove(index);
            list.remove(index);
            return index;
        }
    }
    return -1;
}
I. Draw the variables, references, and objects that exist after the following code executes. Draw node objects as shown below and boxes for variables. The example has all instance variables set to null. The example does not show any of the variables that actually refer to the node object. You must show all variables and all references in your drawing. Use arrows to show references and a forward slash to indicate variables that store null. Assume the node class is the singly linked node from the linked list examples we did in class and that the fields of the class are all public.

```
data   next
\[/ \]/

Node<Object> n1 = new Node<> (null, null); // params are (data, next)
Node<Object> n2 = new Node<> ("AT", new Node<Object> (null, n1));
n1.data = n2.data;
```

Answer:

---

J. The following method takes 10 seconds to complete when each list has a size of 25,000. What is the expected time for the method complete when each list has a size of 50,000?

```
public static int j (LinkedList <Integer> t1, ArrayList<Integer> t2) {
    int r = 0;
    for (int i = 0; i < t1.size(); i++) {
        for (int j = i + 1; j < t2.size(); j++) {
            if (t1.get(i) == t2.get(j))
                r++;
        }
    }
    return r;
}
```
K. How could the order (Big O) of method j from the previous question be reduced? In other words, what change could we make to the code to reduce the order without changing the functionality of the method from the client's perspective? You cannot change the parameter signature or create new objects.

L. The following values are inserted 1 at a time in the order shown to an initially empty binary search tree using the simple algorithm demonstrated in class. Draw the resulting tree.

```
37 73 12 20 101 12 0 20
```

M. What is the result of a post order traversal of a tree produced in question 1.L?

```
_____________________
```

N. What is the minimum number of nodes in a full binary tree with a height of 4?

```
_____________________
```

O. The following method takes 0.01 seconds to complete when \( \text{hm.size()} = 1,000 \). What is the expected time for the method to complete when \( \text{hm.size()} = 1,000,000 \)? BST is a binary search tree class and it uses the simple add algorithm presented in class.

```java
public static BST<Integer> o(HashMap<Integer, String> hm) {
    BST<Integer> t = new BST<>();
    for (Integer i : hm.keySet()) {
        t.add(i);
    }
    return t;
}
```
2. Linked Lists 1 (17 points) - Complete the rangeEqualsTarget instance method for the LinkedIntList class. The method returns true if the sum of the elements in a given range equal a given target.

- You may not use any other methods in the LinkedIntList class unless you implement them yourself as a part of your solution.
- The LinkedIntList class uses singly linked nodes.
- The list has a reference to the first node in the chain of nodes. If the list is empty, first stores null.
- The last node in the chain of nodes has its next reference set to null.
- You may use the nested IntNode class.
- You may not use any other Java classes or native arrays.

```java
public class LinkedIntList {
    // Refers to first node in structure. Stores null if list empty
    private IntNode first;

    private static class IntNode {
        // The nested IntNode class.
        private int data;
        private IntNode next;
    }
}
```

Examples of calls to rangeEqualsTarget (int tgt, int start, int stop) on various lists. rangeEqualsTarget is abbreviated as ret in these examples.

The range is from start inclusive to stop exclusive. The precondition is 0 <= start <= stop, however the range may extend passed the end of the list. If the list does not have the full range of elements specified by the parameters start and stop, the method returns true if the sum of the elements that are present in the list as specified by the range equal the target value.

- [] . ret(0, 0, 5) -> returns true
- [] . ret(-3, 0, 5) -> returns false
- [0, 5, 0, 5, 1] . ret(0, 1, 4) -> returns false
- [0, 5, 0, 5, 1] . ret(10, 1, 4) -> returns true
- [0, 5, 0, 5, 1] . ret(0, 2, 3) -> returns true
- [0, 5, 0, 5, 1] . ret(0, 3, 4) -> returns false
- [0, 5, 0, 5, 1] . ret(0, 3, 3) -> returns true
- [0, 5, 0, 5, 1] . ret(5, 3, 3) -> returns false
- [0, 5, 3, 5, 1, 8] . ret(17, 2, 5) -> returns false
- [0, 5, 3, 5, 1, 8] . ret(17, 2, 6) -> returns true
- [0, 5, 3, 5, 1, 8] . ret(12, 2, 6) -> returns false
- [0, 5, 3, 5, 1, 8] . ret(17, 2, 10) -> returns true
- [0, 5, 3, 5, 1, 8] . ret(17, 10, 12) -> returns false
- [0, 5, 3, 5, 1, 8] . ret(0, 10, 12) -> returns true
Do not create any new node objects in your solution, only IntNode variables.
Do not use recursion in your answer.

/** pre: 0 <= start <= stop
   post: Per the problem description. */
public boolean rangeEqualsTarget (int tgt, int start, int stop) {
3. Linked Lists 2 (17 points) - Complete the `combineIgnoreValue` instance method for the `LinkedIntList` class. The method creates and returns a new `LinkedIntList` object with elements equal to the sum of elements at the same index in the calling object and other list passed as a parameter. However, if either of the nodes contain a target value, no value is added to the result for the elements at that index.

The method uses the same class as question 3 with the same implementation details.

- You may use the provided 0 argument constructor.
- You may not use any other methods in the `LinkedIntList` class unless you implement them yourself as a part of your solution.
- You may use the nested `IntNode` class.
- You may not use any other Java classes or native arrays.

```java
public class LinkedIntList<E> {
    private IntNode first;

    public LinkedIntList() { first = null; }

    private static class IntNode {
        // The nested IntNode class.
        private int val;
        private IntNode next;
    }
}
```

Examples of calls to `LinkedIntList combineIgnoreValue(LinkedIntList other, int tgt)`

<table>
<thead>
<tr>
<th>this:</th>
<th>[3, 5, 8, 5]</th>
</tr>
</thead>
<tbody>
<tr>
<td>other:</td>
<td>[6, 2, 10, -3, 10, 11]</td>
</tr>
<tr>
<td>tgt = 7</td>
<td></td>
</tr>
<tr>
<td>returned list:</td>
<td>[9, 7, 18, 2, 10, 11] // okay for 7 to appear as sum</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>this:</th>
<th>[3, 7, 8, 7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>other:</td>
<td>[6, 2, 10, 7, 10, 7]</td>
</tr>
<tr>
<td>tgt = 7</td>
<td></td>
</tr>
<tr>
<td>returned list:</td>
<td>[9, 18, 10]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>this:</th>
<th>[]</th>
</tr>
</thead>
<tbody>
<tr>
<td>other:</td>
<td>[6, 2, 10, 7, 10, 7]</td>
</tr>
<tr>
<td>tgt = 7</td>
<td></td>
</tr>
<tr>
<td>returned list:</td>
<td>[6, 2, 10, 10]</td>
</tr>
</tbody>
</table>

Do not use recursion in your answer.
/* pre: other != null, post: per the problem description. 
Neither list is altered by this method call. */
public LinkedList combineIgnoreValue (LinkedList other,
        int tgt) {

4. **Stacks (17 points)** Implement a method, `sameBottomN`, that determines if the bottom N elements of two stacks are the same. The method is not part of the Stack class. The method returns true if the bottom N elements of two stacks passed as parameters are equal, false otherwise. If either or both stacks have less than N elements the method returns false.

The stack class has the following methods and constructor:

```java
boolean isEmpty(), void push(E val), E top(), E pop(),
Stack() // creates empty Stack
```

The precondition to `top` and `pop` is `!isEmpty()`. The precondition to `push` is `val != null`.

You may create and use up two temporary Stacks.

You may use the Stack methods and constructor listed above and the `equals` method from the `Object` class. Do not use any other Java classes or methods. Do not use recursion.

Examples of calls to `static <E> boolean sameBottomN(Stack<E> s1, Stack<E> s2, int n)`. In these examples the Stacks are holding `Integer` objects.

Given `s1` and `s2` above for these examples:

```java
s1 top -> 4 4 <- s2 top
  2 3 3
  3 5
  5
```

sameBottomN(s1, s2, 1) -> returns true
sameBottomN(s1, s2, 2) -> returns true
sameBottomN(s1, s2, 3) -> returns false
sameBottomN(s1, s2, 4) -> returns false
sameBottomN(s1, s2, 5) -> returns false
sameBottomN(s1, s2, 6) -> returns false

Given `s1` and `s2` above for these examples:

```java
s1 top -> 4 4 <- s2 top
  2 2
```

sameBottomN(s1, s2, 1) -> returns true
sameBottomN(s1, s2, 2) -> returns true
sameBottomN(s1, s2, 3) -> returns false
/* pre: s1 != null, s2 != null, n >= 1
   post: per the problem description. s1 and s2 are restored to their original state by the end of this method. */
public static <E> boolean sameBottomN(Stack<E> s1, Stack<E> s2, int n)
5. Binary Trees - (17 points.) Complete a helper method for a binary tree class that alters the binary tree so that it is a full binary tree. Recall a full binary tree is one in which every node is a leaf or has 2 children. A full binary tree does not have any internal nodes with a single child. An empty tree is considered a full binary tree. The binary tree class for this question stores chars, but it is not a binary search tree.

The helper method changes the binary tree to a full binary tree by adding nodes to internal nodes that have a single child. The method returns the number of nodes added to the tree.

Consider the following examples:

<table>
<thead>
<tr>
<th>original tree</th>
<th>tree after method call</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>/ \</td>
<td>/ \</td>
</tr>
<tr>
<td>B C</td>
<td>B C</td>
</tr>
<tr>
<td>/ / \</td>
<td>/ / \</td>
</tr>
<tr>
<td>D E F</td>
<td>D B E F</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>G</td>
<td>G E</td>
</tr>
</tbody>
</table>

In this example notice how the internal nodes that stored B and E have had a child added. The value in the new child shall be the same as the parent node. The method would return 2.

<table>
<thead>
<tr>
<th>original tree</th>
<th>tree after method call</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>/ \</td>
<td>/ \</td>
</tr>
<tr>
<td>I J</td>
<td>I J</td>
</tr>
<tr>
<td>/ \</td>
<td>/ \</td>
</tr>
<tr>
<td>K L</td>
<td>K L</td>
</tr>
</tbody>
</table>

No change, already a full binary tree. Method returns 0.

<table>
<thead>
<tr>
<th>original tree</th>
<th>tree after method call</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Q</td>
<td>Q P</td>
</tr>
</tbody>
</table>

Method returns 2.

```java
public class BinaryTree {  // The Binary Tree class for this question.
    private BNode root;  // root == null iff tree is empty

    private static class BNode {
        private char data;
        private BNode left;
        private BNode right;
        public BNode(char ch) { data = ch; }
    }
}
```

Complete the `makeFull` for the `BinaryTree` class. You may use the `BNode` class. Do not use any other methods from the `BinaryTree` class unless you implement them as part of your answer. Do not create any data structures or use any other Java classes or methods.
/* pre: none  
   post: Per the problem description. */
public int makeFull() {

6. Recursion (17 Points) Write a recursive backtracking method that given a tree (not a binary tree), finds the path length (number of links or edges) between a start value and a destination value. If no path exists, the method returns a value less than 0. The nodes in the tree store an int and a list of child nodes.

The tree does not store duplicate values, so the start and end values will be present at most once in the tree.

Consider the following example. The root node stores 5. The root node has 4 child nodes.

```
          5
         /|
        /  |
       3  12 10 7
        /|     |
       6 -3 11 42 19 17 -5 21
       /|     |
      13 2 8 151 -13 23 37 73 36 52 28 -19
      /|     |
     89 90 98
     /|
    92 225
```

Consider these calls to the IntTree class instance method, int pathLength(int start, int dest)

- pathLength(5, 500) -> returns value < 0, no path present, 500 not in tree
- pathLength(200, 10) -> returns value < 0, no path present, 200 not in tree
- pathLength(5, 10) -> returns 1
- pathLength(10, 5) -> returns value < 0, no path present,
- pathLength(12, 2) -> returns 2
- pathLength(13, 8) -> returns value < 0, no path present
- pathLength(10, 19) -> returns value < 0, no path present
- pathLength(7, 225) -> returns 4

Use the following IntTree class and the nested TNode class.

```
public class IntTree {
    private TNode root; // root == null iff tree is empty

    private static TNode {
        private int data;
        private List<TNode> children; children == null iff leaf node
        // No elements of children are null.
    }

    You may use methods from the List interface, but do not create any new data structures or TNode objects. Do not use any other Java classes or methods. For this question only, do no create any other helper methods.
```
/* pre: start != dest
   post: return the path length from the node in this IntTree that contains start to the node in this IntTree that contains dest if they exist and dest is in a node that is a descendant of the node that contains start. If no path exists, return a value < 0. */

public int pathLength(int start, int dest) {
    return help(start, dest, root, -1);
}

private int help(int start, int dest, TNode n, int distanceFromStart){