Your exam shall be scanned. Please write DARKLY and NEATLY.
Please PRINT your name and UTEID clearly in the boxes provided.

Your Name:

Your UTEID:

Instructions:

1. Write your exam number on pages 3, 5, 7, 9, and 11.
2. There are **4** questions on this test. 100 points available.
3. You have 2 hours to complete the test.
4. Place your final answers on this test. **NO SCRATCH PAPER SHALL BE ACCEPTED FOR GRADING. You cannot add pages to the exam.**
5. **This exam is 100% closed**. You may not use **outside resources of any kind** while taking the test. No calculators, mobile devices, electronic devices, or notes of any kind.
6. When answering coding questions, ensure you follow the restrictions of the question.
7. Do **not** write code to check the preconditions.
8. On coding questions 2 and 3, you **may** implement your own helper methods.
9. On coding questions make your solutions as efficient as possible given the restrictions of the question.
10. **Test proctors shall not address 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.**
11. When you complete the test show the proctor your UTID and give them the test. Please place used and unused scratch paper in the appropriate boxes at the front of the room. Please leave the room quietly

1. (2 points each, 50 points total) Short answer. Place your answer **in the box** 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. Recall, when asked for Big O your answer shall be the most restrictive correct Big O function. Closest without going under.
   e. Assume $\log_2(1{,}000) = 10$ and $\log_2(1{,}000{,}000) = 20$.

A. What is returned by the method call `a(4)`?

```
public static int a(int x) {
    if (x == 0) {
        return 2;
    }
    return (x * 2) + a(x - 1);
}
```

B. What is returned by the method call **`mosTQ(5)`**?

```
public static int mosTQ(int x) {
    if (x <= 2) {
        return 3;
    }
    return x + mosTQ(x - 2) + mosTQ(x - 1);
}
```

C. What is returned by the method call **`c(2634)`**? *Courtesy of Jaxon*

```
public static int c(int n) {
    if (n == 0) {
        return 0;
    } else if (n % 2 == 0) {
        return n % 10 - c(n / 10);
    }
    return n % 10 + c(n / 10);
}
```

D. What is the order (Big O) of method **`c`** from 1.C? N = **`n`**

E. What is output by the following code? The code uses the **`java.util.HashMap`** class. *Courtesy of Jaxon*

```
Map<Integer, Integer> m = new HashMap<>();
for (int i = 1; i <= 130; i++) {
    m.put(i % 7, i);
    m.put(i % 11, i);
    m.put(i % 13, i);
}
System.out.print(m.size());
```

F. What is output by the following code? The code uses the **`java.util.TreeMap`** class. Recall the format of the **`String`** returned by Java map's **`toString`** method.
`{key1=value1, key2=value2, ..., keyN=valueN}`

```
Map<Integer, Integer> m2 = new TreeMap<>();
int[] data = {5, 2, 0, 0, 5, 5, 2};
for (int i = 0; i < data.length; i++) {
    m2.put(data[i], i);
}
System.out.print(m2);
```

G. The following method takes 2 seconds to complete when **`map.size() =`** 200,000 and **`list.size() =`** 50,000. What is the expected time for the method to complete when **`map.size() =`** 600,000 and **`list.size() =`** 100,000? **`list`** is a **`java.util.LinkedList`**. Assume every element in **`list`** is present as a key in **`map`**. *Courtesy of Alisha*

```
public static int g(HashMap<String, Integer> map,
                    LinkedList<String> list) {
    int total = 0;
    for (String val : list) {
        if (map.containsKey(val)) {
            total += map.get(val);
        }
    }
    return total;
}
```

H. The following method takes 2 seconds to complete when **`list.size()`** = 10,000. What is the expected time for the method to complete when **`list.size()=`** 30,000. **`list`** is a **`LinkedList314`** as implemented in lecture. In each case 50% of the elements in **`list`** are **`>= tgt`**.

```
public static int h(LinkedList314<Integer> list, int tgt) {
    int total = 0;
    for (int i = 0; i < list.size(); i++) {
        if (list.get(i) >= tgt) {
           total += list.get(i);
        }
    }
    return total;
}
```

I. The following method takes 10 seconds to complete when **`n`** = 1,000,000. What is the expected time for the method to complete when **`n`** = 2,000,000? *Courtesy of Nidhi*,

```
public static int i(int n) {
    int t = 0;
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < i; j++) {
            t += i * j;
        }
    }
    return t;
}
```

EXAM # _____

J. Consider the following method. **`list`** is **already sorted**. Of the sorting algorithms we studied which one will **typically** lead to the fewest computations if used by method **`sort`**? Pick the letter of the correct answer. *Courtesy of Leul*

```
public static void j(ArrayList<Integer> list) {
    Random r = new Random();
    list.add(r.nextInt());
    list.add(r.nextInt());
    sort(list);
}
```

A. Selection Sort B. Insertion Sort C. Radix Sort D. Quicksort E. Mergesort

K. The following method takes 10 seconds to complete when **`map.size()`** = 1,000,000 and **`data.length =`** 1,000,000. What is the expected time for the method to complete when **`map.size() =`** 4,000,000 and **`data.length =`** 2,000,000? **`map`** is a **`java.util.TreeMap`**. In each case 50% of the elements in **`data`** are values in **`map`**.

```
public static int k(TreeMap<Integer, String> map, int[] data) {
    int total = 0;
    for (int i = 0; i < data.length; i++) {
        if (map.containsKey(data[i])) {
            total += map.get(data[i]).length();
        }
    }
    return total;
}
```

L. What is output by the following code? The stack is the same the one we implemented in lecture, with an added **`size`** method that returns the number of elements currently in the stack.

```
Stack314<Integer> st = new Stack314<>();
for (int i = 5; i >= 0; i--) {
    st.push(i);
}
int sum = 0;
for (int i = 0; i < st.size(); i++) {
    sum += st.pop();
}
System.out.print(sum);
```

M. Which of the following can be used as the internal storage container for a stack such that all of the four stack operations discussed in lecture are average case O(1)? Answer with the letter of the correct answer.

A. **`LinkedList314`**, the singly linked list from lecture
B. **`java.util.LinkedList`**
C. **`java.util.ArrayList`**
D. a native array
E. None of A through D can provide average case O(1) for the 4 stack operations
F. All of A through D can provide average case O(1) for the 4 stack operations

N. What is output by the following code? The queue is the same as the one we implemented in lecture.

```
    Queue314<Integer> q = new Queue314<>();
    int[] d1 = {5, 3, 2, 7, 1, 2, 4, 12, 3, 9, 8, 3};
// Doesn't follow style guide but not a compile error or logic error.
    for (int x : d1)
        if (x % 2 != 0)
            q.enqueue(x);
    int s = 0;
    for (int i = 0; i < 5; i++) {
        s += q.dequeue();
    }
    System.out.print(s + " " + q.front());
```

O. The following method takes 3 seconds when **`list.size()`** = 250,000. What is the expected time for the method to complete when **`list.size()`** = 500,000. In each case 25% of the elements in **`list`** are **`<= target`**. **`list`** is a **`java.util.LinkedList`**.

```
    public static void o(LinkedList<Integer> list, int target) {
        Iterator<Integer> it = list.iterator();
// Doesn't follow style guide but not a compile error or logic error.
        while (it.hasNext())
            if (it.next() <= target)
                it.remove();
    }
```

P. We have an N by N 2d array of **`int`**s. There are no duplicates. If we traverse the 2d array in row-major order, left to right, top to bottom, the **`int`**s are in sorted ascending order. How efficient can we make the search, using the algorithms presented in lecture, to determine if a value is present in the 2d array without creating any new data structures?
Pick the letter of the correct answer.

A. O(logN) B. O($(\log N)^2$) C. O(N) D. O(NlogN) E. O(N^2)

EXAM # _____

Q. We have an array that initially contains the following numbers:

```
[9, 54, 5, 18, 78, 14, 23, 20, 0, 12]
```

We perform the mergesort algorithm on the array as described in lecture. What will the contents of the array be before the final merge operation? Pick the letter of the correct answer.

A. `[5, 9, 18, 54, 78, 0, 12, 14, 20, 23]`

B. `[0, 5, 9, 14, 18, 20, 23, 54, 78, 12]`

C. `[9, 12, 5, 0, 14, 78, 23, 20, 18, 54]`

D. `[9, 54, 5, 18, 78, 14, 23, 20, 0, 12]`

E. `[20, 0, 12, 23, 54, 14, 5, 18, 78, 9]`

R. The following method takes 1 second to complete when **n** = 20. What is the expected time for the method to complete when **n** = 23?

```
public static int r(int n) {
    if (n <= 0) {
        return 2;
    }
    int t = n - 1;
    return n + r(t) + r(t) + r(t);
}
```

S. What is output by the following code? *Courtesy of Sam*

```
int[] d = {4, 5, 2, 1};
ArrayList<Integer> as = new ArrayList<>();
for (int x : d) {
    as.add(x);
}
System.out.print(s(as.iterator()));

public static int s(Iterator<Integer> it) {
    int s = 0;
    if (it.hasNext()) {
       s += it.next();
       s += s(it);
       s += s(it);
    }
    return s;
}
```

T. We have two methods, **`quicksort`** and **`mergesort`**, that accept an **`int`** array and return a sorted copy of the array using the respective algorithms as demonstrated in lecture. The quicksort implementation picks the middle element of a subarray as its pivot. **`int[] nums`** has 5,000,000 distinct integers in random order. Which of the following expressions is likely to execute faster?
Pick the letter of the best answer. *Courtesy of Sam*

A. **`quicksort(mergesort(nums))`**

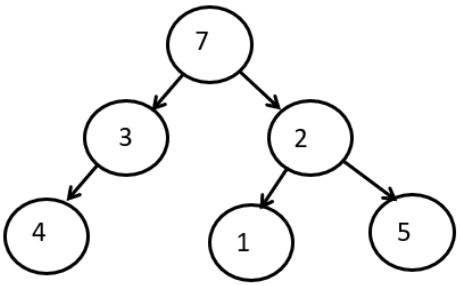
B. **`mergesort(quicksort(nums))`**

C. The two expressions will execute in roughly the same amount of time.

U. In a ***full*** binary tree with 9 nodes what is the minimum possible number of leaf nodes with a depth of 2?

V. In a ***complete*** binary tree with 37 nodes how many nodes have 2 children? *Courtesy of Karnika*

W. Consider the binary tree to the right. We perform a **preorder** traversal on the binary tree and maintain a result initially equal to 0. The value in the first node visited is **added** to the result. The value in the second node is **subtracted** from the result. The value in the third node visited is **added** to the result. The value in the fourth node visited is **subtracted**. This pattern of altering adding and subtracting the values in the nodes continues until the traversal is complete.
What is the result after completing a **pre-order** traversal?
Your answer shall be a single integer.



X. Given the same binary tree from 1.W and the alternate adding - subtracting algorithm of values in nodes when the node is visited, what is the result of an **in-order** traversal of the binary tree?
Your answer shall be a single integer.

Y. Given the same binary tree from 1.W and the alternate adding - subtracting algorithm of values in nodes when the node is visited, what is the result of a **post-order** traversal of the binary tree?
Your answer shall be a single integer.

EXAM # _____

2. **Maps.** (16 points) Complete a method, **`playedMostRanked`**, method as described below.

The method accepts a parameter, a **`Map<String, ArrayList<String>>`** that contains information about college volleyball teams. The keys are **`String`**s, the names of the teams, and the values are **`ArrayList<String>`,** the names of the other teams the team represented as the key has played. Only a portion of the map is shown.

Texas	[A&M, Pitt, SMU, TCU, Stanford, Baylor, Georgia, Vanderbilt, Wisconsin]
A&M	[Texas, SMU, Florida, Alabama, Arkansas, Missouri, TxSt, Trinity, UNT]
SMU	[Texas, A&M, TCU, UNT, Texas Tech, Colorado, Utah, Houston, West Virginia]
UNT	[A&M, SMU, Sam Houston, TCU, TxSt, West Texas, Baylor, Texas Tech, UTD]
TxSt	[UNT, Trinity, UTSA, UTD, Nebraska, Minnesota, Austin, SLU, Northwestern]

The method is also passed a **`Set<String>`** that contains the names of the top 25 ranked volleyball teams.

The method returns the name (String) of the team, that is a key in the map, that has played the most ranked teams. If there is a tie, return any of the team names that are tied for playing the most ranked teams.

You may assume at least one of the teams represented by the keys in the map has played at least one ranked team.

You may use the following methods:

`Map<K, V>` interface: **`Set<E> keySet(), V get(Object key),`**

`Set<E>` interface: **`boolean contains(E value)`**

`Iterator<E>` interface: **`boolean hasNext(), E next()`**

`ArrayList<E>` class: **`E get(int index), int size()`**

You may use for-each loops.

Do not use any other Java methods or classes. Do not create any new data structures. Of course, you may use primitives. **Do not use recursion on your answer.**

```
/* pre: teams != null, no null key,
          none of the elements of the value lists are null
        ranked != null, ranked.size() == 25
   post: per the problem description.
         Neither teams or ranked is altered by this method. */
public static String playedMostRanked(
                 Map<String, ArrayList<String>> teams,
                 Set<String> ranked)
```

Complete the method on the next page.

```
public static String playedMostRanked(
                  Map<String, ArrayList<String>> teams,
                  Set<String> ranked) {
```

EXAM # _____

3. Linked Lists (17 points) Complete the **`addIfFrequencyLessThan`** for the **`LLL314`** class. The method adds a given element to the **end** of the **`LL314`** if there are fewer than N copies of the element already present. The method returns **`true`** if an element was actually added, **`false`** otherwise. Consider these examples with lists of **`String`**s. Added elements underlined.

```
[A, B, C, A, B, B, X].addIfFrequencyLessThan(A, 1) -> returns false
calling object unchanged, not less than 1 A already in the list

[A, B, C, A, B, B, X].addIfFrequencyLessThan(A, 2) -> returns false
calling object unchanged, not less than 2 A's already in the list

[A, B, C, A, B, B, X].addIfFrequencyLessThan(A, 3) -> returns true
calling object becomes [A, B, C, A, B, B, X, A]

[A, B, C, A, B, B, X].addIfFrequencyLessThan(A, 4) -> returns true
calling object becomes [A, B, C, A, B, B, X, A]

[A, B, C, A, B, B, X].addIfFrequencyLessThan(M, 1) -> returns true
calling object becomes [A, B, C, A, B, B, X, M]

[A, B, C, A, B, B, X].addIfFrequencyLessThan(M, 12) -> returns true
calling object becomes [A, B, C, A, B, B, X, M]

[].addIfFrequencyLessThan(M, 1) -> returns true
calling object becomes [M]

[].addIfFrequencyLessThan(M, 12) -> returns true
calling object becomes [M]
```

- **You may not use any other methods in the `LL314` class unless you implement them yourself as a part of your solution. You may not add instance or class variables to `LL314`.**
- The list only has a reference to the first node in the chain of nodes. No size, no last.
- When the list is empty, **`first`** stores **`null`**.
- The list does **not** store **`null`** values. (Hooray!)
- If the list is not empty, the last node in the structure stores **`null`** in its **`next`** variable.
- You may use the nested **`Node`** class and the **`equals`** method on objects.
- **You may not use any other Java classes or native arrays. Do not create ANY new data structures. You may, of course, create a new Node object if necessary.**
- **Do not use recursion in your answer.**

```
public class LL314<E> {
     private Node<E> first; // Stores null if this list is empty.

     // The nested Node class.
     private static class Node<E> {
          private E data;
          private Node<E> next; // Set to null if last node.

          public Node(E val) { data = val; } // Can use!
     }
}
```

```
    // pre: tgt != null, freq >= 1
    // post: Per the problem description.
    public boolean addIfFrequencyLessThan(E tgt, int freq) {
```

EXAM # _____

4. **Recursive Backtracking** (17 points) Complete a helper method that uses recursive backtracking to determine if the words in a given dictionary, stored in an **`ArrayList<String>`**, can be used to form at least one ***word loop*** with exactly **`goal`** number of words. Unlike the anagrams assignment the word loop must have **exactly `goal`** words. For this question, fewer than **`goal`** words is not a valid word loop.

For this question, a word loop is a sequence of words. We can only use words from the given **`dictionary`**. **The first word may be any valid word from the dictionary.** Subsequent words must meet the following condition: The first letter of the next word matches the last letter of the previous word. And the last words last letter must match the first letter of the first word. This forms a word loop as opposed to a word chain.

Consider the following word loop that meets the requirement of a goal of **exactly** 5 words:

`[blast, tall, lab, bad, dub]`

The t in tall matches the t in blast, the l in lab matches the l in tall, the b in bad matches the b in lab, the d in dub matches the d in bad and the b in blast matches the b in dub.

Note, we do not allow words to be reused so this is **not** a valid word loop: (pip and pup appear more than once)

`[pup, pip, pup, pip, pup]`

Complete the following helper method that determines if the Strings in the given **`dictionary`** can be used to form at least one word loop given the requirements above.

```
public static boolean canForm(ArrayList<String> dictionary,
                              ArrayList<String> loop, int goal) {
```

- **You may not add static variables. You may not add new parameters to the `canForm` method.**
- **The only helper method you may add is a helper that determines if the last character in one `String` equals the first character of another `String`. You are not required to add this helper, but you may find it useful. <u>Do NOT add any other helper methods.</u>**
- **`dictionary`** contains the valid words we can use to form the word loop.
- Do not alter **`dictionary`** in any way. Not even temporarily.
- No elements of **`dictionary`** are **`null`**.
- All elements of **`dictionary`** have a **`length()`** >= 2
- Assume the **`ArrayList<String> loop`** is initially empty. In other words, the client that calls **`canForm`** will send a reference to an **`ArrayList<String>`** with **`size() == 0`** for the parameter **`loop`**.
- Assume **`goal`** >= 2. This is the number of words required to complete the word loop. The loop must have exactly **`goal`** words.
- If a word loop can be formed from the given dictionary the **`ArrayList<String> loop`** holds the first loop found when all calls to **`canForm`** are completed. If a word loop cannot be formed from the given dictionary, then **`loop`** shall be empty when all calls to **`canForm`** are completed.
- Do not create any new data structures. No new arrays or lists or maps or sets.
- You may use the following **`String`** methods: **`charAt(int index)`** and **`length()`**
- You may use the following **`ArrayList`** methods: **`int size(), E get(int index), boolean contains(E value), add(E value), remove(int index).`**
- **You may use a for-each loop to iterate through the elements of `dictionary`.**
- **Do NOT use any other Java classes or methods. You can of course use primitive `chars`.**

```
public static boolean canForm(ArrayList<String> dictionary,
                              ArrayList<String> loop, int goal) {
```

If you need extra room for a coding question answer, place it on this page.
Make it clear which question you are answering, 2, 3, or 4.