| Points off | 1 | 2 | 3 | 4 | | | Raw Points Off |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Your Name: ______________________________________

Your UTEID: ______________________________________

Circle your TA's Name: **Anna** **Brad** **David** **Emma** **Justin** **Lilly** **Natalee** **Pavan** **Pranav** **Skyler**

Instructions:

1. There are **4** questions on this test. 100 points available. Scores shall be scaled to 250 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**. Exams not completed in pencil are not eligible for a regrade. You may use highlighters on the exam.
4. **This exam shall be entirely your own work.** You may not use **outside resources of any kind** while taking the test. Please remove any smart watches and put them and any mobile devices away.
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 2 and 3 (but not 4), you may implement your own helper methods.
8. On coding questions make your solutions as efficient as possible given the restrictions of the question.
9. Test proctors will 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.
10. When you complete the test show the proctor your UTID and give them the test. Please place used and used 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 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. Recall when asked for Big O your answer shall be the most restrictive correct Big O function. For example, Selection Sort is average case $O(N^{2)}$, but per the formal definition of Big O it is correct to say Selection Sort is $O(N^3)$ , $O(N^4)$ and so forth.
Give the most restrictive, correct Big O function. (Closest without going under.)
e. Assume `log₂(1,000) = 10` and `log₂(1,000,000) = 20.`

A. What is returned by the method call **`a(25)`?** ________________________

```
public static int a(int n) {
    if (n <= 0)
        return 1;
    return a(n / 2) + 1;
}
```

B. What is the order of method **`a`** from question 1.A? N = the parameter **`n`**. ____________

C. What is output when the following code is run? ______________________________

```
c(1);

public static int c(int n) {
    if (n >= 10) {
        System.out.print('+');
        return 2;
    }
    System.out.print(n);
    int temp = 2 + c(n * 2 + 1);
    System.out.print(temp);
    return temp;
}
```

D. What is output when the following code is run? ______________________________

```
System.out.print(d("COMPUTER").length());

public static String d(String s) {
    if (s.length() <= 4)
        return "-";
    return "++" + d(s.substring(1)) + d(s.substring(2));
}
```

E. The following code takes 1 second to complete when **`n`** = 500,000. What is the expected time for the method to complete when **`n`** = 2,000,000 ? Assume the **`Random.nextDouble`** method is O(1). The linked list class is the one developed in lecture.

__________________

```
public static LinkedList314<Double> e(int n, Random r) {
    LinkedList314<Double> result = new LinkedList314<>();
    for (int i = 0; i < n; i++)
        result.insert(0, r.nextDouble()); // position, value
    return result;
}
```

F. The following method takes 5 seconds to complete when **`n`** = 10,000. What is the expected time for the code to complete when **`n`** = 20,000? Assume the **`Random.nextDouble`** method is O(1). The method returns a **`java.util.ArrayList`**.

__________________

```
public static ArrayList<Double> f(int n, Random r) {
    ArrayList<Double> result = new ArrayList<>();
    for (int i = 0; i < n; i++)
        result.add(0, r.nextDouble()); // position, value
    return result;
}
```

G. What is the worst-case order of the following method if **`list`** is a **`java.util.LinkedList`** object? N = **`list.size()`**. **`List`** is the **`java.util.List`** interface.

__________

```
public static int g(List<Integer> list, int tgt) {
    int s = 0;
    int i = list.size() - 1;
    while (s < tgt && i >= 0) {
        s += list.get(i);
        i--;
    }
    return i;
}
```

H. What is the worst-case order of method **`g`** above, if **`list`** is a **`java.util.ArrayList`** object.? N = **`list.size()`**

__________

I. The following method takes 5 seconds to complete when **`list.size()`** = 1,000,000. What is the expected time for the method to complete when **`list.size()`** = 2,000,000 ? Assume the **`Math.random()`** method is O(1) and recall it returns a value between [0.0, 1.0) with a uniform distribution. The parameter list is a **`java.util.LinkedList`** object.

__________

```
public static int i(List<Integer> list) {
    int r = 0;
    Iterator<Integer> it = list.iterator();
    while (it.hasNext()) {
        int t = it.next();
        if (Math.random() < 0.25) {
            r += t;
            it.remove();
        }
    }
    return r;
}
```

J. When method **`i`** is passed a **`java.util.ArrayList`** and **`list.size()`** = 10,000 the method takes 2 seconds to complete. What is the expected time for the method to complete when **`list.size()`** = 30,000 ?

______________________________

K. What is output by the following code? _____________________________________________

```
int[] data = {5, -2, 7, 0};
HashMap<String, Integer> h = new HashMap<>();
String s = "";
for (int i = 0; i < data.length; i++) {
    s += (i + 2);
    h.put(s, data[i]);
}
System.out.print(h);
```

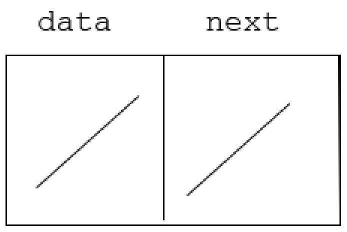
L. The following method takes 5 seconds to complete when **`list.size() =`** 1,000. The search method uses the binary search algorithm as demonstrated in class with an updated parameter of **`LinkedList314`**. What is the expected time for the method to complete when **`list.size() =`** 2,000? Assume the target is not present in either case.

__________

```
public static int indirectSearch(LinkedList314<Integer> list,
                                 int tgt) {
    return search(list, tgt);
 }
```

Consider the following **`Node`** class for questions M and N.

```
public class Node {
    public Object data;
    public Node next;
}
```



M. What is output by the following code? ____________________

```
Node n1 = new Node();
Node n2 = new Node();
n1.next = n2;
n2.next = n1;
System.out.print(n1.next.next.next.data);
```

N. Consider the following code. Draw the variables, references, and objects that exist after the following code executes. Draw node objects as shown above (to the right of the **`Node`** class code) and boxes for variables. The example has all instance variables set to **`null`**.

```
Node n1 = new Node();
Node n2 = new Node();
n2.next = n1;
n1.data = new int[2];
n2.data = n1.data;
```

O. Consider the following array. It is sent to a sort method. The array is shown at an intermediary step during the sorting algorithm. Of the 5 sorts we discussed in lecture (selection, insertion, radix, quick, merge) which algorithm does the method most likely use?

____________________

```
[99, 321, 89, 512, 73, 617, 181] // original array

[512, 617, 321, 73, 181, 89, 99] // array at intermediary step
```

P. Consider the following list of **`String`**s. It is sent to a sort method with the **`Comparator`** shown. The array is shown after the sorting is complete. Of the 5 sorts we discussed in lecture (selection, insertion, radix, quick, merge) which algorithm or algorithms does the sort method most likely use?

____________________________________________

```
["Java", "Lisp", "C++", "Tk", "Flex", "C"] // original array

public class LenComp implements Comparator<String> {
    public int compare(String s1, String s2) {
        return s1.length() - s2.length();
    }
}

["C", "Tk", "C++", "Flex", "Lisp", "Java"] // sorted array
```

Q. When we covered the stack data structure in lecture we discussed postfix expressions. What does the following postfix expression evaluate to? Single integer for answer.

```
-2 6 17 12 - + *
```

_____________

R. The following code takes 1 second to complete. What is the expected time for the method to complete when the length of the new array is 2,000,000 instead of 1,000,000?
The sort method uses the insertion sort algorithm as demonstrated in class.

_____________

```
    int[] data2 = new int[1_000_000];
    sort(data2);
```

S. What is output by the following code? The queue class is a fair, unbuffered queue.

_______________________________________

```
Queue314<Integer> q = new Queue314<>();
int x = 2;
for (int i = 0; i < 4; i++) {
    x = x * 2 - 1;
    q.enqueue(x);
}
String s2 = "";
while (!q.isEmpty()) {
    s2 += q.front() + " ";
    q.dequeue();
}
System.out.print(s2);
```

T. What is output by the following code? The stack class is the one implements in lecture.

____________________________________________________

```
Stack314<Integer> st2 = new Stack314<>();
for (int i = 0; i < 10; i++)
    st2.push(i);
String s = "";
for (int i = 0; i < 10; i += 3)
    s += st2.pop() + " ";
System.out.print(s);
```

U. In a complete binary tree with 20 nodes, how many nodes in the tree have a depth of 3? ______

V. The following values are added in the order shown, left to right, to an initially empty binary search tree using the simple algorithm demonstrated in class. What is the height of the root node of the resulting tree?

**`12, 20, 18, 35, 40, -5, 37`** ______

W. The following values are added in the order shown, left to right, to an initially empty binary search tree using the simple algorithm demonstrated in class. What is the result of a post order traversal of the resulting tree?

**`5, -7, 3, 5, 5, 3, 9, 7`** ____________________________________

X. The following method takes 1 second to complete when **n** = 10,000. What is the expected time for the method to complete when **n** = 20,000. The **BST314** class uses the simple add algorithm demonstrated in class.

_________________________

```
public static BST314<Integer> x(int n) {
    BST314<Integer> result = new BST314<>();
    for (int i = 0; i < n; i++) {
        result.add(i);
    }
    return result;
}
```

Y. 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. The **BST314** class uses the simple add algorithm demonstrated in class. Assume **Random.nextInt()** is O(1).

_________________________

```
public static BST314<Integer> x(int n, Random r) {
    BST314<Integer> result = new BST314<>();
    for (int i = 0; i < n; i++) {
        result.add(r.nextInt());
    }
    return result;
}
```

**Extra Credit:** The UT Women's Volleyball Team won the national championship fall of 2022. How many times in total has the University of Texas Women's Volleyball team won the national championship?

______________

2. **Linked Lists.** (13 points) Complete the `lastIndexOf` instance method for the `LinkedList314` class. The method determines the index in the list of the last occurrence of a given target value. **Do not use recursion in your answer.**

- **<u>You may not use any other methods in the LinkedList314 class unless you implement them yourself as a part of your solution.</u>**
- The `LinkedList314` class uses singly linked nodes.
- 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 lists do not store `null` values.
- If the list is not empty, the last node's `next` reference stores `null`.
- 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 use recursion in your answer.**

```
public class LinkedList314 <E> {

     // refers to first node in the chain of nodes.
     private Node<E> first;

     // The nested Node class.
     private static class Node<E> {
          private E data;
          private Node<E> next;
     }
}
```

Examples of calls to `lastIndexOf(E tgt)`.
In the examples below the elements of the list are `String` objects.

```
[A, B, A, C, A, D].lastIndexOf(A) -> returns 4

[A, B, A, C, A, D].lastIndexOf(D) -> returns 5

[A, B, A, C, A, D].lastIndexOf(S) -> returns -1

[].lastIndexOf(A) -> returns -1

[S, K, L, K, L, M, L].lastIndexOf(S) -> returns 0

[S, K, L, K, L, M, L].lastIndexOf(L) -> returns 6
```

Complete the `lastIndexOf` method on the next page.

```
/* pre: tgt != null
   post: return the index of the last occurrence of tgt in this list
         or -1 if tgt is not present in this list. This list is not
         altered by this method call.
*/
public int lastIndexOf(E tgt) {
```

3. **Maps and Sets** (20 points) Complete a method that returns the highest similarity score that exists between a pair of books and based on the characters in the books. The map given to this method consists of keys that are **`String`**s, representing the titles of books, and values that are **`Set`**s of **`String`**s, representing the characters in the book.

Consider this example map (keys on the left, values on the right):

| | |
|---|---|
| The Wonderful Wizard of Oz | {Dorothy, Toto, Scarecrow, Tinman, Lion, The Wizard, Glinda} |
| The Marvelous Land of Oz | {Tip, Mombi, Jack, Scarecrow, Tinman, General Jinjur} |
| Ozma of Oz | {Uncle Henry, Dorothy, Billina, Tik-Tok, Ozma, Nome King} |
| Dorothy and the Wizard in Oz | {Dorothy, Zeb, Eureka, Jim, The Wizard} |
| The Road to Oz | {Dorothy, Toto, Shaggy Man, Button-Bright, Polychrome} |

The similarity score for a pair of books is the number of characters the books have in common divided by the total number of characters in the two books, excluding duplicates.

For example, *The Wonderful Wizard of Oz* and *The Marvelous Land of Oz* have two characters in common: Scarecrow and Tinman. They have 11 total characters excluding duplicates: Dorothy, Toto, Scarecrow, Tinman, Lion, The Wizard, Glinda, Tip, Mombi, Jack, General Jinjur.

The similarity score for this pair of books is 2 / 11 = 0.1818…

You may use the following methods and classes:

From the **`Map`** interface:
```
public Set<K> keySet()
public V get(K key)
public V put(K key, V val)
public V remove(K key)
```

From the **`Set`** interface:
```
public Iterator<E> Iterator
public boolean contains(Object o)
public int size()
```

You may use all methods from the **`Iterator`** interface either explicitly or implicitly.

You may use the **`equals`** method on **`Object`**s.

**Do not use any other Java classes or methods other than those listed above.**

**Do not create any new data structures.**

**Do NOT use recursion in your solution.**

**The map of books is not altered by this method.**

```
/* pre: books != null, books.size() >= 2
   post: per the problem description. books is not altered by this method.
*/
public static double highestSimilarity(Map<String, Set<String>> books) {
```

4. **Recursive Back Tracking** (17 points) Complete the `minSteps` method as described below. The method accepts a 2d array of `int`s representing elevations on a 2d area of land. The value at each element represents the elevation of that point on land. The higher the elevation, the higher the number, the lower the elevation the lower the number. We are also given position of the starting cell, two integers specifying the row and column.

In this problem we can move to adjacent cells in one of four directions, up, down, left, and right **if the difference between the elevations of the two cells is less than or equal to 10.**

The `minSteps` method returns the minimum number of steps required to move from the starting location to the edge of the area. The edge of the area is a row or column than is on the edge of the 2d array.

If it is not possible to move from the starting location to the edge the method returns `Integer.MAX_VALUE / 2`.

Consider this example area. All values in areas are guaranteed to be greater than or equal to 0 and less than or equal to 1,000,000. No negative values in the 2d array.

| | | | | | | |
|---|---|---|---|---|---|---|
| 99 | 75 | 95 | 80 | 90 | 90 | 90 |
| 99 | 85 | 80 | 75 | 70 | 70 | 65 |
| 99 | 50 | 45 | 75 | 75 | 75 | 75 |
| 99 | 99 | 40 | 35 | 30 | 25 | 75 |
| 10 | 80 | 84 | 40 | 75 | 20 | 75 |
| 40 | 99 | 99 | 50 | 75 | 25 | 10 |

If we start at row 1, col 1, (the 85) there are multiple paths to the edge, but the shortest has a length of 1, simply moving up to the 75 on the edge.

If we start at row 2, col 1 (the 50) there are again multiple paths to the edge, but the shortest has a length of 5 (right to 45, down to 40, right to 35, down to 40, down to 50).

If we start at row 4, col 1 (the 80) there is no way to reach the edge. **Recall, we can only move to cells above, below, left, and right with a difference of no more than 10.**

Complete the `minSteps` method below.

**You may NOT add any other helper methods, class variables, or instance variables.**

**You may NOT alter the parameters of the `minSteps` method. (Cannot add new parameters.)**

**HINT: You MAY make temporary changes to the 2d array of ints, but must undo those changes to restore the 2d array of its to its original state by the time the method completes.**

**Do not use any other Java methods or classes except the provided `rc_deltas` class constant and the `Integer.MAX_VALUE` constant. You may of course use the public `length` field for arrays.**

**Do not create any new data structures.**

**Your solution must use recursive backtracking.**

```
// Class constant to change row and column
private static final int[][] rc_deltas = {{1, 0, -1,  0},
                                          {0, 1,  0, -1}};

/* pre: area != null, area.length > 0, area[0].length > 0, area is
rectangular (all rows have the same number of columns, all elements of
area are in the range [0, 1_000_000], row and col are in bounds.
  post: per the problem description. */
public static int minSteps(int[][] area, int row, int col) {
```