Points off	1	2	3	4			Total off	Net Score

Your Name: ______________________________________

Your UTEID: ______________________________________

Circle your TA's Name: **David K.** **David T.** **Elizabeth** **Henry** **Lilly** **Nina** **Pranav** **Skyler** **Sam** **Trisha**

Instructions:

1. There are **4** questions on this test. 100 points available. Scores will 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**.
4. You may not use a calculator or **outside resources of any kind** while taking the test.
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.
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. (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. Using the techniques and rules from lecture, what is the T(N) of the following method? N = data.length

____

```
public static int a(int[] data) {
    int r = 0;
    for (int i = 0; i < data.length; i++) {
        r += data[i] * i;
        for (int j = 0; j < data.length; j++) {
            r += data[j] / 2;
        }
        data[i] /= 2;
    }
    return r;
}
```

B. What is the worst-case order of method b? Assume method `check` is always O(N) and method `process` is always O(1). N = `data.length`.

____

```
public static ArrayList<Integer> b(int[] data) {
    ArrayList<Integer> result = new ArrayList<>();
    for (int i = 0; i < data.length; i++) {
        if (check(data[i])) {
            process(result, data[i]);
        } else {
            result.add(0, data[i]); // position, value
        }
    }
    return result;
}
```

C. What is the order of method c? N = the parameter n. ____________________

```
public static int c(int n) {
    int r = 0;
    for (int i = 1; i <= n; i += 2) {
        for (int j = i; j > 0; j--) {
            r += i % j;
        }
    }
    for (int i = 1; i < n; i++) {
        r += i * i;
    }
    return r;
}
```

D. The following method takes 20 seconds to complete when `data.length` = 1,000,000. What is the expected time for the method to complete when `data.length` = 2,000,000?

____

```
private static ArrayList<int[]> d(int[] data) {
    ArrayList<int[]> r = new ArrayList<>();
    for (int x : data) {
        int[] t = new int[10];
        t[0] = x;
        r.add(0, t); // insert method, (position, value)
    }
    return r;
}
```

E. The following method takes 5 seconds to complete when `data.length` = 20,000. What is the expected time for the method to complete when `data.length` = 40,000. Assume all elements of data are between 1000 and 9999 inclusive.

____

```
public static String e(int[] data) {
    String r = "";
    for (int x : data) {
        r += " ";
        r += x;
    }
    return r;
}
```

F. What is output by the following code? ______________________________________

```
    ArrayList<String> list = new ArrayList<>(3);
    list.add("A");
    list.add(1, "B");
    list.add(1, "C");
    list.add(1, "D");

    list.set(2, "E");

    list.add(2, "F");
    System.out.println(list);
```

G. A method is O(N^3). It takes the method 1 second to complete when N = 10,000. How long do you expect the method to take when N = 30,000?

____

H. What is output by the following method if the list sent to the method initially contains `[5, 3, -2, 0, 12]`?

____

```
public static void h(ArrayList<Integer> list) {
    Iterator<Integer> it = new Iterator<>(list);
    it.next();
    it.next();
    while (it.hasNext()) {
        System.out.print(it.next() + " ");
    }
}
```

I. A method is O(2^N). It takes the method 0.1 seconds to complete when N = 40. How long do you expect the method to take when N = 45?

____

J. Given the class named `Sam` below, what is the output of the given client code?

____

```
public class Sam {
    private int[] data;
    public Sam(int[] d) { data = d; }
    public void add(int i) { data[i]++; }
}

// client code
int[] nums = {6, 3, 0, 7};
Sam s = new Sam(nums);
nums[2] += 5;
s.add(1);
s.add(3);
nums[1] += 3;
System.out.println(Arrays.toString(nums))
```

K. Given the class named `Test` below, what is the output of the given client code?

____

```
public class Test {
    private int x;
    public Test(int x) { this.x = x;}
    public String toString() { return "" + x; }
    public static void show() {
        System.out.print(this);
    }
}

// client code
Test t = new Test(5);
Test.show();
```

L. Using the techniques and rules from lecture, what is the T(N) of the following method? N = the parameter n.

____

```
    public static int tn2(int n) {
        int t = 0;
        final int LIMIT2 = n * 2;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < LIMIT2; j++) {
                int temp = i * j;
                t += temp;
            }
        }
        return t;
    }
```

M. What is output by the following program when it is run? ________________________________

```
public class QM {
    public static void main(String[] args) {
        QM m = new QM();
        System.out.print(m.toString());
    }
}
```

N. What is output by the following code? ________________________________

```
ArrayList<Integer> list = new ArrayList<>();
Object obj = list;
list.add(3);
list.add(1);
list.add(4);
System.out.println(obj);
```

O. Why doesn't the following code compile?

____

```
ArrayList<Integer> list = new ArrayList<>();
Object obj = list;
System.out.println(obj.size());
```

P. What is the worst-case order of the following method? N = `list.size()`. ________________

```
public static void p(ArrayList<String> list, int x) {
    for (int i = list.size() - 1; i >= 0; i--) {
        if (list.get(i).length() > x) {
            list.remove(i);
        }
    }
}
```

Q. What is output by the following code? ___________________________________

```
ArrayList<Object> list = new ArrayList<>();
list.add(list.toString());
list.add(2.72);
list.add(new ArrayList<Integer>());
System.out.println(list.toString());
```

R. A method is O($N\log_2 N$). It takes the method 10 seconds to complete when N = 1,000,000. How long do you expect the method to take when N = 2,000,000?

____

For questions S through Y on the following page, refer to the classes on page 7.

S. What is output by the following client code? (Not in `PieceOfFurniture` the class.)

____

```
PieceOfFurniture pf = new PieceOfFurniture();
pf.numLegs += 2;
System.out.println(pf);
```

T. What is output by the following client code? (Not in `PieceOfFurniture` the class.)

____

```
PieceOfFurniture pf2 = new PieceOfFurniture();
pf2.addColor("Red");
pf2.addColor("Tan");
System.out.println(pf2.numColors());
```

U. What is output by the following client code? ___________________________________

```
PieceOfFurniture pf3;
pf3 = new Desk(6, 4, "Brown");
System.out.println(pf3.toString());
```

V. What is output by the following client code? ___________________________________

```
Desk d1 = new Desk(6, 2, "Black");
Desk d2 = d1;
d2.addColor("Orange");
d2.addDrawer();
System.out.println(d1);
```

W. What is output by the following client code? ___________________________________

```
PieceOfFurniture pf4;
pf4 = new Desk(0, 3, "Black");
pf4.addDrawer();
System.out.println(pf4);
```

X. What is output by the following client code? ___________________________________

```
Desk d7 = new Desk(0, 6, "Black");
d7.addColor("White");
System.out.println(d7.numColors());
```

Y. For the each of the following lines of code state if they would compile or not.

```
Desk d8 = new Object(); // Compiles? ____________________

Desk d9 = new Desk();   // Compiles? ____________________
```

For questions S through Y, refer to the following classes.

```
public class PieceOfFurniture {
    private ArrayList<String> colors;
    private int numLegs;

    public PieceOfFurniture() { numLegs = 4; }

    public PieceOfFurniture(int n, String color) {
        numLegs = n;
        colors = new ArrayList<>();
        colors.add(color);
    }

    public String toString() { return numLegs + ""; }
    public void addColor(String color) {
        colors.add(color);
    }
    public int numColors() { return colors.size(); }
}

public class Desk extends PieceOfFurniture
        implements Comparable<Desk> {
    private int numDrawers;

    public Desk(int d, int legs, String color) {
        super(legs, color);
        numDrawers = d;
    }

    public String toString() {
        return numColors() + " " + numDrawers;
    }
    public void addDrawer() {numDrawers++;}

    public int compareTo(Desk d) {
        return this.numDrawers - d.numDrawers;
    }
}
```

2. **Lists.** (17 points) To demonstrate encapsulation and the syntax for building a class in Java, we developed a **`GenericList`** class that can store elements of any data type. Recall our **`GenericList`** class stores the elements of the list in the first size elements of a native array of **`Object`**s. An element's position in the list is the same as the element's position in the array. The array may have extra capacity and thus may be larger than the list it represents. **The list does not allow client code to add null elements.**

Complete an instance method for the **`GenericList`** class named **`getEqualFronts`**. The method accepts a second **`GenericList`** as an explicit parameter. The method creates and returns a new **`GenericList`**. The elements in the returned list are those from the beginning of the calling list and the other list that are at the same position and equal to each other based on the equals method. **The resulting list shall contain all the equal elements from the start of the two lists up to, but not including, the first two elements that do not equal each other.**

```
/*   pre: other != null
     post: Per the problem description. This list and other are not
     altered by this method. */
public GenericList<E> getEqualFronts(GenericList<E> other) {
```

Examples of calls to the **`getEqualFronts`** method. (The values shown are String objects).

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

The GenericList class:

```
public class GenericList<E> {
     private E[] con;
     private int size;

     public GenericList() {
          con = (E[]) new Object[10];
     }
```

You may not use any methods from the GenericList class other than the given constructor unless you implement them as a part of your solution.

Do not use any other Java classes or methods except the equals method and native arrays.

```
/*   pre: other != null
     post: Per the problem description. This list and other are not
     altered by this method. */
public GenericList<E> getEqualFronts(GenericList<E> other) {
```

3. (17 points) Write an instance method for the **`Names`** class from assignment 3 that returns an **`ArrayList<String>`** of the names (**`Strings`**) in the **`Names`** object that are ***one hit wonders***. A one hit wonder is defined to be a name that is unranked, then is ranked better than some cutoff, and then in the next decade goes to being unranked again. Recall ***better ranks are small. 50 is better (more popular) than*** 700. The resulting **`ArrayList<String>`** only has one copy of a name even if it meets the criteria for a on hit wonder multiple times.

Examples. If the required rank was 750, here are some of names that meet the criteria.

Prince 789 660 719 910 880 0 0 **0 747 0** 913

Sunshine 0 0 0 0 0 0 **0 732 0** 0 0

Mcarthur 0 0 0 **0 669 0** 0 0 0 700 0 (only add Mcarthur once)

Brigitte 0 0 0 0 0 **0 613 0** 859 0 0

Aron 868 **0 663 0** 0 855 773 548 525 577 618

The following names are examples that **do not** meet the one hit wonder criteria if the required rank was 750.

Cedrick 0 0 0 0 0 0 0 693 804 0 0 (not unranked after jumping to 693

Zion 0 0 0 0 0 0 0 0 0 0 300 (not unranked after jumping to 300)

Alexandra 728 0 0 0 817 778 581 314 85 28 36 (not unranked before jumping to 728)

Spring 0 0 0 0 0 0 0 750 0 0 0 (not better than the required rank of 750)

The **`Names`** class for this question:

```
public class Names {
    private ArrayList<NameRecord> records;
    // All NameRecords in this name object have NUM_DECADES ranks.
    private final int NUM_DECADES;

    /* pre: requiredRank < 1000
       post: per the problem description */
    public ArrayList<String> getOneHitWonders(int requiredRank) {
```

You may use the following two methods from the `NameRecord` class:

```
int getRank(int decade) returns the rank for the given decade.
Uses 0 based indexing. 0 -> first decade, 1 -> second decade, ...
NUM_DECADES - 1 -> last decade.  Returns 0 if unranked in the given decade.

String getName() returns the name of this NameRecord.
```

You may not add methods to the `NameRecord` class.

You may use the `E get(int pos)`, `int size()`, and `add(E val)` methods from the `ArrayList` class. You may call the zero argument `ArrayList` constructor to create a single `ArrayList<String>`.

Do not use any other Java classes or methods.

```
/* pre: requiredRank < 1000
   post: per the problem description */
public ArrayList<String> getOneHitWonders(int requiredRank) {
```

4. Other Data Structures (16 points) Implement a **`removeAll`** method for an **`ArraySet`** class.

Recall, a set does not allow duplicate values and from the client's perspective, the elements are not in any particular order. This means after calling method that has the potential to alter the set (a mutator method), the relative order of the elements in the set may be altered from the client's perspective.

The **`ArraySet`** class stores its element in a native array. There may be extra capacity in the array. The elements are stored in the first **`size`** elements of the internal array. The ArraySet does no allow **`null`** to be an element of the set, but elements in the array that is the internal storage container (**`con`**) that are not currently storing an element of the set, the extra capacity elements in the array, **DO** store **`null`**.

Write a method, **`removeAll`** that removes from the calling **`ArraySet`** all elements that are present in another **`ArraySet`** that is sent as a parameter. The method returns the number of elements removed by the method.

Consider these following examples. The elements shown in this example are **`String`**s. -> indicates what the calling **`ArraySet`** becomes after the method is complete.

```
(A, B, Z, K, L).removeAll((Z, M, A)) -> (B, K, L), returns 2
```

Realize that from the client's perspective the order of the elements after this call to **`removeAll`** is complete could be (K, B, L) instead of (B, K, L). From the client's view, after this call to **`removeAll`** is completed, the order of the element in this set could be any permutation of (B, K, L).

```
(A, L, Z, B, K).removeAll(()) -> (A, L, Z, B, K) , returns 0

().removeAll((A, L, Z, B, K)) -> (), returns 0

(A, L, Z, B, K).removeAll((AL, SO, JJ)) -> (A, L, Z, B, K) , returns 0

(A, L, Z, B, K).removeAll((K, L)) -> (A, Z, B), returns 2

(AL, L, GO).removeAll((GO, K, L, AL, J, A)) -> (), returns 3
```

Here is the **`ArraySet`** class for this question:

```
public class ArraySet<E> {
     private int size; // number of elements in this set
     private E[] con;
```

Do not use any Java methods or classes other than the Object equals method.

Do not create any new arrays or other data structures in your solution. O(1) space.

```
/* pre: other != null
   post: per the problem description. other is not altered. */
public int removeAll(ArraySet<E> other) {
```