Points off	1	2	3	4			Raw Points Off

Your Name: _____

Your UTEID: _____

Circle your TA's Name: **Aman** **Bersam** **Brayden** **Casey** **Devon**
Eliza **Gracelynn** **Lauren** **Namish** **Nidhi** **Pavan**

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. You may not use a calculator or **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, 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 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.**
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 $\log_2(1{,}000) = 10$ and $\log_2(1{,}000{,}000) = 20$.

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 t = 0;
    final int LIMIT = data.length;
    for (int i = 0; i < LIMIT; i += 4) { // Note the update statement.
        t += data[i] / 2;
        t += data[i] * data[i];
    }
    return t;
}
```

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

_____

```
public static double b(int n) {
    double t = 0.0;
    final int LIMIT = n * n;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            t += i / (j + 1.0);
        }
        for (int j = 0; j < LIMIT; j++) {
            t += i * j;
        }
    }
    return t;
}
```

C. What is the order of the following code? N = **data.length**. Method **process** is O(N) where N is the length of the array sent as an argument.

_____

```
public static int c(int[] data) {
    int t = 0;
    for (int i = 0; i < data.length / 2; i++) {
        t += process(data, i);
        t += data[i];
    }
    return t;
}
```

D. What is the order of the following code? N = **data.length**. Method **process2** is O(N) where N is the length of the array sent as an argument. Method **check1** is O(N) where N is the length of the array sent as an argument. Assume **check1** returns **true** for the given array roughly half of the time.

_____

```
public static int d(int[] data) {
    int t = 0;
    for (int i = 0; i < data.length; i++) {
        t += process2(data, i);
        for (int j = 0; j < data.length; j++) {
            if (check1(data, j)) {
                t += process2(data, j);
            }
         }
    }
    return t;
}
```

E. A method is $O(N^3)$. The method takes 1 second to complete when N = 10,000. What is the expected time for the method to complete when N = 30,000?

_____

F. A method is O(N!). The method takes 1 second to complete when N = 99. What is the expected time for the method to complete when N = 101?

_____

G. The following code takes 0.2 seconds to complete when **`list.size()`** = 1,000,000. What is the expected time for the code to complete when **`list.size()`** = 2,000,000?
The code uses the **`java.util.ArrayList`** class.

_____

```
public static int g(ArrayList<Integer> list) {
    int result = 0;
    for (int i = 0; i < list.size(); i++) {
        for (int j = 1; j < list.size(); j *= 2) {
            result += list.get(i) * list.get(j);
        }
    }
    return result;
}
```

H. What is the worst-case order of the following code? N = **`list.size()`**. _____

```
public static void h(ArrayList<Double> list, double target) {
    for (int i = list.size() - 1; i >= 0; i--) {
        if (list.get(i) < target) {
            list.remove(i); // Remove element at given position.
        }
    }
}
```

I. What is output by the following code? The code uses the **`java.util.ArrayList`** class.

_____

```
ArrayList<Integer> list = new ArrayList<>(10);
list.add(3);
list.add(0, 5); // position to add, value
list.add(0, 1);
list.add(2, 4);
list.add(1, 7);
list.remove(3); // Remove element at given position.
System.out.print(list);
```

J. What is output by the following code when run? _____

```
public static void j(ArrayList<Integer> list) {
    list.add(3);
    list.add(0);
    list = new ArrayList<Integer>();
    list.add(4);
    System.out.print(list.size() + " ");
}

// client code
ArrayList<Integer> listJ = new ArrayList<>();
listJ.add(12);
j(listJ);
System.out.print(listJ.size());
```

K. The following method takes 5 seconds to complete when **n** = 1,000,000. What is the expected time for the method to complete when **n** = 2,000,000? The code uses the **GenericList** method developed in lecture and the **resize** method for the **GenericList** class is as shown.

_____

```
public static GenericList<Integer> j(int n) {
    GenericList<Integer> r = new GenericList<>();
    for (int i = 0; i < n; i++) {
        r.add(i * i);
    }
    return r;
}

// resize method in the GenericList class
private void resize() {
    con = Arrays.copyOf(con, con.length + con.length / 3 + 1);
}
```

L. What is output by the following code when run? _____

```
String s1 = "ABC";
String s2 = "MNOP";
String r = "";
if (s1.compareTo(s2) >= 0)
    r += "x";
else
    r += "o";
r += s1.length() == s2.length();
System.out.print(r);
```

M. Which of the following lines of code, if any, cause compile errors? _____

```
ArrayList listM = new ArrayList(20); // 1
listM.add(12); // 2
listM.add("Devon"); // 3
listM.add(listM.get(1)); // 4
listM.add(listM.get(10)); // 5
listM.add(listM.get(1).substring(3)); // 6
listM.add(new ArrayList<Double>()); // 7
```

N. What is output to the screen when method **n** is called with a **java.util.ArrayList** that contains the following values?

_____

```
["Devon", "Gracelynn", "Brayden", "Aman", "Pavan", "Lauren", "Namish",
     "Nidhi", "Lauren", "Eliza", "Bersam"]

    public static void n(ArrayList<String> list) {
        int t = 0;
        Iterator<String> it = list.iterator();
        for (int i = 0; i < 6; i++)
            it.next();
        while (it.hasNext())
            if (it.next().length() >= 6)
                t++;
        System.out.print(t);
    }
```

O. What is output by the following code when it run? _____

```
    public class Book implements Comparable<Book> {
        private int pages;

        public Book(int p) {
            pages = p;
        }

        public int compareTo(Book other) {
            return pages - other.pages;
        }
    }

    // client code
    Comparable b1 = new Book(10);
    Comparable b2 = new Book(5);
    System.out.print(b1.compareTo(b2));
```

For questions 1.P - 1.Y refer to the classes at the end of the exam.
You may detach that page from the exam.

P. For each of the following circle if the code compiles or causes a compile error. (1 point each)

```
ElectricCar ec = new Car();          // compiles       compile error

Vehicle v1 = new Motorcycle("red"); // compiles       compile error
```

Q. For each of the following circle if the code compiles or causes a compile error. (1 point each)

```
Object o3 = new ElectricCar();   //    compiles       compile error

Motorcycle mv = new Car();       //    compiles       compile error
```

R. What is output by the following code? _____

```
Vehicle vr = new ElectricCar();
vr.recharge();
System.out.print(vr);
```

S. What is output by the following code? _____

```
Motorcycle mc3 = new Motorcycle("red");
System.out.print(mc3.getTopSpeed());
```

T. What is output by the following code? _____

```
Vehicle v4 = new Car();
v4.enhance();
System.out.print(v4);
```

U. What is output by the following code? _____

```
ElectricCar ec3 = new ElectricCar();
System.out.print(ec3.getTopSpeed());
```

V. Yes or No, will the following code compile? If **not**, explain why not.

_____

```
Motorcycle mv4 = new Motorcycle("orange");
System.out.print(mv4.getClass());
```

W. What is output by the following code? _____

```
Object o1 = new Motorcycle("black");
Object o2 = new Motorcycle("black");
System.out.print( (o1 == o2) + " " + o1.equals(o2));
```

X. What is output by the following code? _____

```
Vehicle v5 = new Vehicle();
v5.enhance();
v5.enhance();
System.out.print(v5.getTopSpeed());
```

Y. Assume the following method is added to the **`Vehicle`** class. What is output by the client code?

_____

```
// Added to the Vehicle class.
public static void checkDoors(Car c) {
    if (c.doors == 4) {
        c.doors = 6;
    }
}

// client code
Car c4 = new Car();
Vehicle.checkDoors(c4);
System.out.print(c4);
```

2. **Lists.** (14 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. This version of our **`GenericList`** class stores the elements of the list in the first **`size`** elements of a native array of **`E`**'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.

Create an instance method for the **`GenericList`** class **`removeLast`**. The method removes the last occurrence of a target value from the list if present. If not, present the list is unaltered. The method returns the index of the last occurrence of the target value before it is removed. If the target was not present in the list the method returns -1.

Consider these examples with lists of Strings.

`[A, B, AA, C, A, B].removeLast(A) ->` returns 4, list becomes `[A, B, AA, C, B]`

`[A, B, AA, C, A, B].removeLast(B) ->` returns 5, list becomes `[A, B, AA, C, A]`

`[A, B, AA, C, A, B].removeLast(FX) ->` returns -1, list unaltered

`[AAA, B, AA, C, A, B].removeLast(AAA)` -> returns 0, list becomes `[B, AA, C, A, B]`

`[].removeLast(FX) ->` returns -1, list unaltered

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

```
public class GenericList<E> {

   // This GenericList does NOT allow the client to store null values.

    private E[] con;
    private int size;
```

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

You may call the `equals` method on objects. Of course, you may use the `length` field for arrays.

You may not use any other Java classes or methods.

Do not create any new data structures.

Complete the method on the next page.

```
// pre: target != null
// post: Per the problem description.
public int removeLast(E target) {
```

3. Baby Names (18 points) Complete an instance method in the **`Names`** class from assignment 2 that returns an **`ArrayList<String>`** of all the names that are ***formerly popular***. Formerly popular names are names that are **not** ranked in the last N decades. And, the name is more popular than some cutoff value in at least one decade when it **is** ranked. Consider the following examples with a requirement of not being ranked in at least the **last 4** decades, but being ranked **at least 300 or better** in the one of the decades preceding that. (In this database the names are ranked in 9 decades each.)

```
Seymour, ranks: 638 324 223 409   0 0 0 0 0 (not ranked the last 5)
Tillie,  ranks:   0 287 295   0 900 0 0 0 0 (not ranked in the last 4)
Lulu,    ranks: 292 463 135   0   0 0 0 0 0 (not ranked in the last 6)
```

Given the criteria of not ranked in the last 4 decades and a rank of 300 or better (lower) in at least one decade, the following names would NOT meet the requirement.

Omer is ranked in one of the last 4 decades (990 third from end).

```
Omer,    ranks: 357 466 284 734  0  0  990  0  0
```

Ophelia is not ranked the last 5 decades, **but** never has a rank better than or equal to 300.

```
Ophelia, ranks: 370 344 393 482  0  0  0  0  0
```

If no names meet the criteria return an empty **`ArrayList<String>.`**

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

```
public class Names {

     /* The NameRecords in this Names object.
        All NameRecords have the same number of decades. */
     private ArrayList<NameRecord> names;
}
```

Methods you may use from **`NameRecord`**: **You may not add methods to the `NameRecord` class.**

```
String getName() - return the name for this NameRecord
int numDecades() - return the total number of decades, including unranked
int getRank(int decade) - return the rank for the given decade. Uses 0
based indexing. Returns 0 if unranked in the given decade.
```

From the **`ArrayList`** class:

```
ArrayList() - construct an empty ArrayList
add(E obj) - add obj to the end of this ArrayList
int size() - number of elements in this ArrayList
E get(int pos) - access element at given position
```

You may also use **`ArrayList`**s as the target of **`for-each`** loops.

Do not use any other Java classes or methods besides those listed above.
Create a single `ArrayList<String>`, the result to return. Do not create any other new data structures.

```
/* pre: 1 <= minRank < 1000, numUnrankedEnd < the number of decades
        NameRecords in this Names object are ranked.
  post: this Names object is not altered and per the problem
        description. */
public ArrayList<String> formelyPopular(int minRank,
                                        int numUnrankedEnd) {
```

4. **Other Data Structures** (18 points) Complete the **`boolean isSubset(MultiSet<E> other)`** instance method for a **`MultiSet`** class. A **`MultiSet`** or Bag allows duplicate values, like lists, but does not maintain the elements in a particular order from the client's perspective, like sets.

A client may add values in the following order **`(B, A, B, C, A, A, B, A)`** to a **`MultiSet`**, but we can store the values internally in any order we want.

So internally we could store **`(B, B, B, A, A, A, A, C)`**.

We can simplify this further by storing each distinct element once and storing the number of times the element appears. **`((B, 3), (A, 4), (C, 1))  -> (element, frequency)`**

```
public class MultiSet<E> { // Does NOT allow client to add null.

    private ValueAndFrequency<E>[] con; /* May have extra capacity.
      ValueAndFrequency objects stored in first numDistinct spots.*/

    private int numDistinct; /* The number of distinct elements in
             this MutliSet. In the above example numDistinct is 3.*/

    private int size; /* The total number of elements in this
      MultiSet, including duplicates. In the above example size is 8. */

    private static class ValueAndFrequency<E> {
        private E element;     // Never null.
        private int frequency; // Always >= 1
    }
}
```

Complete the **`boolean isSubset(MultiSet<E> other)`** instance method for the MultiSet class. The method returns **`true`** if **`other`** is a subset of the calling object, **`false`** otherwise.

Examples of calls to **`isSubset`**:

```
[B, B, B, C, C, C, C, A].isSubset([A, C, C]) -> returns true
[B, B, B, C, C, C, C, A].isSubset([C, C, A]) -> returns true
[B, B, B, C, C, C, C, A].isSubset([A, A, C]) -> returns false
[B, B, B, C, C, C, C, A].isSubset([]) -> returns true
[B, B, B, C, C, C, C, A].isSubset([A, B, M]) -> returns false
[].isSubset([]) -> returns true
[].isSubset([A, C, C]) -> returns false
```

Do not use any other Java methods or classes except the `MultSet` instance variables and the nested `ValueAndFrequency` class. You may use the equals method on `Objects`.

Of course, you can use the native arrays **`length`** field. **Do not create any new data structures.**

```
/* pre: other != null
   post: per the problem description. */
public boolean isSubset(MultiSet<E> other) {
```

For questions P through Y, refer to the following classes. You may detach this page from the exam.

```
public class Vehicle {
    private int topSpeed;

    public Vehicle(int s) { topSpeed = s; }

    public void enhance() { topSpeed += 5; }

    public int getTopSpeed() { return topSpeed; }
}


public class Motorcycle extends Vehicle {

    private String color;

    public Motorcycle(String c) {
        super(150);
        color = c;
    }

    public String getString() { return color + getTopSpeed(); }
}

public class Car extends Vehicle {

    private int doors;

    public Car() {
        super(90);
        doors = 4;
    }

    public void enhance() { doors++; }

    public String toString() { return "zoom: " + doors; }
}


public class ElectricCar extends Car {

    private int batteryLife;

    public void recharge() { batteryLife = 4; }

    public String toString() { return "days: " + batteryLife; }
}
```