Your exam shall be scanned. Please PRINT your name and UTEID clearly in the space provided.

Your Name: _____

Your UTEID: _____

Instructions:

1. There are **4** questions on this test. 100 points available.
2. You have 2 hours to complete the test.
3. Place your final answers on this test. **NO SCRATCH PAPER SHALL BE ACCEPTED FOR GRADING. You cannot add pages to the exam.**
4. **This exam is 100% closed**. You may not use **outside resources of any kind** while taking the test. No calculators, mobile devices, electronic devices of any kind, or notes of any kind.
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 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 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. Closest without going under.
   e. Assume $\log_2(1{,}000) = 10$ and $\log_2(1{,}000{,}000) = 20$.

A. A method is $O(N^4)$. The method takes 2 seconds to complete when N = 5,000. What is the expected time for the method to complete when N = 10,000?

_____

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

```
public static int b(int n) {
    int t = 0;
    for (int i = 0; i < n; i++) {
        int t2 = i * i;
        t += t2;
    }
    final int LIMIT = n * 3;
    for (int i = 0; i < LIMIT; i++) {
        t += i * 3;
    }
    return t;
}
```

_____

C. Method **b** from 1.B, takes 2 seconds to complete when **n** = 1,000,000. What is the expected time for the method to complete when **n** = 5,000,000?

_____

D. Using the techniques and rules from lecture, what is the T(N) of the following method? N = **data.length** _____

```
public static int d(int[] data) {
    int t = 0;
    for (int i = 0; i < data.length; i++) {
        int t1 = data[i];
        for (int j = 0; j < data.length; j++) {
            int t2 = data[j];
            for (int k = 0; k < data.length; k++) {
                t += data[k] * t1 / t2;
            }
        }
    }
    return t;
}
```

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

F. What is the order of the following method? N = **n**.
Method **check** is O(N) where N = **n**. *Courtesy of Alisha* _____

```
public static int f(int n) {
    int t = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
             t += check(i, j, n);
        }
    }

    for (int i = 0; i < n; i++) {
        t += check(i, i, n);
    }
    return t;
}
```

G. What is the **best-case** order of the following method?
N = **list.size()**. *Courtesy of Casey* _____

```
public static void g(ArrayList<Integer> list) {
    for (int i = 0; i < list.size(); i++) {
        if (list.get(i) % 2 == 0) {
            list.remove(0); // remove by index
        } else {
            list.remove(list.size() - 1); // remove by index
        }
    }
}
```

H. What us output by the following code?
Courtesy of Samarth _____

```
ArrayList<String> letters = new ArrayList<>();
letters.add("C");
letters.add("CS");
letters.add(1, "B"); // (index, element to add, we called this insert)
letters.add(2, "GO");
letters.remove(1); // (index)
letters.add(2, letters.get(letters.size() - 3));
System.out.println(letters);
```

I. **true or false:** There is no way for a class that states **implements Comparable** to compile **without** implementing a version of the **compareTo** method inside that class.

_____

J. Recall our implementation of the **`IntList`** in class. A student proposed the following implementation of the **`toString`** method. **true or false**: This version of the method meets our requirements for the **`IntList toString`** method.

_____

```
// In the IntList class
public String toString() {
    return Arrays.toString(con);
}
```

K. The following method takes 3 seconds to run given the client code. What is the expected time for the method to run when the line in the client code **`int n = 25_000;`** is changed to **`int n = 50_000;`** ? *Courtesy of Sean*

_____

```
// client code
int n = 25_000;
int[] data = new int[n];
k(data);

public static int k(int[] data) {
    int count = 0;
    for (int i = 0; i < data.length; i++) {
        for (int j = i + 1; j < data.length; j++) {
            if (data[j] > data[i]) {
                for (int k = j + 1; k < data.length; k++) {
                    if (data[k] > data[j]) {
                        count++;
                    }
                }
            }
        }
    }
    return count;
}
```

L. Given the final version of our **`GenericList`** class from lecture which of the following lines of code cause an error at **compile time**? (Meaning we would not be able to run the code.) List the numbers of all lines that give a **compile error** or state, **No compile errors.**

_____

```
GenericList<String> list = new GenericList<>(); // 1
list.add("CS314"); // 2
String s1 = "CS311"; // 3
list.add(s1); // 4
list.add(list.get(0).toLowerCase()); // 5
list.add(s1.substring(1, 15)); // 6
list.add("!");
```

M. Given the final version of our **`GenericList`** class from lecture which of the following lines of code cause an error at **compile time**? (Meaning we would not be able to run the code.) List the numbers of all lines that give a **compile error** or state, **No compile errors.**

_____

```
GenericList<String> list1 = new GenericList<>(); // 1
list1.add(314 + ""); // 2
String s2 = "CS429"; // 3
list1.add(s2.charAt(2)); // 4
list1.add(list1.get(0).substring(1, 2)); // 5
list1.add(439); // 6
list1.add(s2 + s2 + s2); // 7
list1.add(list1.toString()); // 8
```

N. **true or false**: In Java, all data structures can be the ***target*** of for-each loops, also known as the enhanced for loop. In the example below **`data1`** is the ***target*** of the for-each loop.

_____

```
// Toy code to illustrate what we mean by the 'target' of the
// for-each loop. data is the target of the for-each below.
int[] data1 = {5, 7, 12, 5, -3, 14, 17};
for (int x : data1) {
    System.out.print(x);
}
```

For questions 1.O - 1.X refer to the classes at the end of the exam.
Please detach that page from the exam.

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

```
StandardPackage sp1 = new FragilePackage(2, 50); // compiles     error

Object o1 = new Package(5);                       // compiles     error
```

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

```
Package p1 = new InsuredPackage(10, 3);   //    compiles      error

FragilePackage fp1 = new Package(5);      //    compiles      error
```

Q. What is output by the following code? _____

```
FeatherPackage fp2 = new FeatherPackage();
System.out.print(fp2.getCost() + " " + fp2.getWeight());
```

R. What is output by the following code? _____

```
InsuredPackage ip1 = new InsuredPackage(2, 5);
System.out.print(ip1);
```

S. What is output by the following code? _____

```
InsuredPackage ip2 = new InsuredPackage(2, 5);
System.out.print(ip2.getCost());
```

T. What is output by the following client code? _____

```
Package p2 = new FragilePackage(2, 5);
p2.setInsure(5, 2);
System.out.print(p2.getCost());
```

U. What is output by the following code? _____

```
Package p3 = new Package(10);
Package p4 = new Package(10);
System.out.print(p3.equals(p4));
```

V. What is output by the following code? _____

```
Package p5 = new FragilePackage(2, 5);
System.out.print(p5.getCost());
```

W. What is output by the following code? _____
Inspired my Sam

```
public static void w(InsuredPackage ip) {
    ip.setInsure(2);
    System.out.print(ip.getCost() + " ");
    ip = new InsuredPackage(3, 3);
    ip.setInsure(3);
    System.out.print(ip.getCost() + " ");
}

// client code
InsuredPackage ip = new InsuredPackage(1, 1);
w(ip);
System.out.print(ip.getCost());
```

X. Consider the following new class. What is output by the following client code? *Inspired by Casey*

_____

```
public class MediaPackage extends Package {

    public MediaPackage(int weight) {
        super(weight);
    }

    public int getCost(int mult) { return 2 + mult * weight / 2; }
}

// client code
MediaPackage mp1 = new MediaPackage(10);
System.out.print(mp1.getCost(10));
```

Y. Consider the following classes. What is output when the client code is run?

_____

```
public class A {
    public A() { System.out.print(4); }
}

public class B extends A {
    public B() { System.out.print(1); }
}

public class C extends B {
    public C() { System.out.print(3); }
}

// client code
C cObj = new C();
```

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. 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. The extra array elements store **`null`**.

Create an instance method for the **`GenericList`** class **`copyWithoutTarget`**. The method creates and returns a new **`GenericList`** that is a copy of the calling object, except elements equal to a given target value are excluded from the copy. The relative order of the other elements is unchanged.

Consider these examples with lists of Strings.

```
[A, B, C, A, B, B, X].copyWithoutTarget(A) returns [B, C, B, B, X]

[A, B, C, A, B, B, X].copyWithoutTarget(B)  returns [A, C, A, X]

[C, C, C, C, C].copyWithoutTarget(C) returns []

[A, B, C, A, B, B].copyWithoutTarget(X) returns [A, B, C, A, B, B]

[].copyWithoutTarget(G) returns []
```

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;

    public GenericList() { } // Sets instance vars to 0 equivalent.
```

You may not use any other methods or constructors from the `GenericList` class, other than the zero-argument constructor shown, 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 <u>other than</u> a native array for the resulting `GenericList` and of course the resulting `GenericList` itself.

```
// pre: target != null
// post: Per the problem description. This GenericList is not altered.
public GenericList<E> copyWithoutTarget(E target) {
```

3. MathMatrix (16 points) Complete an instance method in the **`MathMatrix`** class from assignment 2 that creates and returns a **`MathMatrix`** that is the result of concatenating one **`MathMatrix`** object to another.

Example:

Calling object, left hand side operand

1	5	3
2	7	6

Explicit parameter, right hand side operand

12	10	-3	0
9	-5	4	8

Result:

1	5	3	12	10	-3	0
2	7	6	9	-5	4	8

Recall from the assignment, our **`MathMatrix`** objects are rectangular: a given **`MathMatrix`** object has the same number of columns in every row.

Also, every **`MathMatrix`** object is at least 1 x 1. We never have **`MathMatrix`** objects with zero rows or zero columns.

```
public class MathMatrix {

     // No extra capacity. No other instance variables.
     private int cells[][];

     public MathMatrix(int rows, int columns) {
         cells = new int[rows][columns];
     }
}
```

You may not use any other methods or constructors from the `MathMatrix` class, other than the constructor shown.

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 <u>other than</u> a 2d native array for the resulting `MathMatrix` and of course the resulting `MathMatrix` itself.

Complete the method on the next page.

```
// pre: rhs != null. The number of rows in this and rhs are equal.
public MathMatrix concatenate(MathMatrix rhs) {
```

4. **Other Data Structures** (17 points) Complete the **`getIntersection(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 is present in the **`MutliSet`**.
`[(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. */

    public MultiSet(int initialCapacity) {  // pre: initialCap > 0
        con = new ValueAndFrequency[initialCapacity]; }

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

        private ValueAndFrequency(E e, int f) {// pre:e != null, f > 0
            element = e; frequency = f; }
    }
}
```

Complete the **`MultiSet<E> getIntersection(MultiSet<E> other)`** instance method for the **`MultiSet`** class. The method creates and returns a new **`MultiSet<E>`** that is the intersection of the calling object and the explicit parameter, **`other`**.

Examples of calls to **`getIntersection`**: The returned value shown is an abstract representation from the client's view. The order of elements in the returned **`MultiSet`** could be different. Internally we are using an array of **`ValueAndFrequency`** objects.

```
[B, B, B, C, C, C, C, A].getIntersection([A, C, C]) -> returns [C, C, A]
[B, B, C, C, C, C, A].getIntersection([Y, Z, X]) -> returns []
[].getIntersection([A, A, C]) -> returns []
[B, B, B, C, A, A].getIntersection([A, A, A, B, B]) ->returns [A, A, B, B]
[B, X, X, C, C, C, C, A].getIntersection([A, B, M]) -> returns [A, B]
[].getIntersection([]) -> returns []
[B, A, X, X, X, C, C].getIntersection([A, A, B, B]) -> returns [A, B]
```

Do not use any other Java methods or classes except the `MultiSet` instance variables, constructor and the nested `ValueAndFrequency` class. You may use the `equals` method on `Objects` and the `Math.min(int, int)` method. Do not create any new data structures besides the new `MultiSet` and the necessary internal variables for that `MultiSet`.

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

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.

For 1.P through 1.Y, refer to the following classes. Detach this page from the exam. *Courtesy of Sean*
Program Hygiene guide NOT followed so this fits on one page.

```
public class Package { // as in package that can be mailed
    private int weight;

    public Package(int weight) {  this.weight = weight;   }

    public int getCost() {  return 5 + weight * 2; }

    public int getWeight() {  return weight; }
}

public class FeatherPackage extends Package {

    public FeatherPackage() {  super(50); }

    public int getWeight() {  return 0; }
}

public class StandardPackage extends Package {
    private int shippingTime;

    public StandardPackage(int weight, int time) {
        super(weight);
        shippingTime = time;
    }

    public int getCost() {  return super.getCost() + 2 * shippingTime; }

    public String toString() {  return "t: " + shippingTime; }
}

public class InsuredPackage extends StandardPackage {
    private int insuranceCost;

    public InsuredPackage(int wt, int t) {  super(wt, t); }

    public void setInsure(int cost) {  insuranceCost = cost; }

    public int getCost() {  return super.getCost() + insuranceCost * 3; }
}

public class FragilePackage extends Package {
    private int fee;

    public FragilePackage(int weight, int fee) {
        super(weight);
        this.fee = fee;
    }

    public int getCost() {  return 10 * fee + getWeight(); }
}
```