Points off	1	2	3	4	5		Total off	Net Score

Your Name: _____

Your UTEID: _____

Circle your TAs Name: Aish Amir Anthony Ethan Fatima
Hailey Jacob Lucas Rebecca Terrell

Instructions:

1. There are **5** questions on this test. 100 points available. Scores will be scaled to 200 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 any other electronic devices 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. (1 point each, 20 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 Order N^2, but per the formal definition of Big O it is correct to say Selection Sort is Order N^3 and Order N^4.
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. What is the T(N) of the following method?
N = `a.length` _____

```
public static int tn(int[] a, int[] b) { // pre: a.length <= b.length
      int r = 0;
      int s = 0;
      for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a.length; j++)
                  if (i != j) {
                        int t = a[i] * b[i];
                        t *= 2;
                        r += t;
                  }
            s++;
      }
      return  r - s;  }
```

B. What is the average case order (Big O) of the following method? N = `list.size()`.
All elements in the list contain between 1 and 10 characters with each length being equally likely.

_____

```
public static String methodB(ArrayList<String> list) {
     String r = "";
     for (int i = 0; i < list.size(); i++) {
          String s = list.get(i);
          if (s.length() % 2 == 0) {
               r += s;
          }
     }
     return r;
}
```

C. What is the worst case order (Big O) of the following method? N = `data.length`. _____

```
public static int methodC(int[] data) {
     int r = 0;
     for (int i = 1; i < data.length; i *= 4)
          for (int j = 0; j < data.length; j++)
               if (data[j] / 2 == data[i])
                    r++;
     return r;
}
```

D. What is the order (Big O) of the following method? N = `data.length`
Method `help` is $O(N^2)$ when N is the length of the array it is sent. _____

```
public static int[] methodD(int[] data) {
     int[] r = new int[data.length * 2];
     for (int i = 0; i < data.length; i++) {
          int t = 0;
          int x = data[i];
          t = help(data, x);
          r[i] = t;
     }
     return r;
}
```

E. A method is $O(N^3)$. It takes 5 seconds to complete when N = 5,000. What is the expected time in seconds for the method to complete when N = 15,000?

_____

F. A method is $O(N^2 \log_2 N)$. It takes 1 second for the method to complete when N = 1,000.
What is the expected time in seconds for the method to complete when N = 2,000?

_____

G. A method is $O(N^2)$. It takes 0.1 seconds for the method to complete when N = 1,000. If we are willing to let the method run for 10 seconds, what would N be? In other words, how many elements can the method process if we allow it to run for 10 seconds?

_____

H. What is output by the following code?

```
ArrayList<String> listH = new ArrayList<>();
listH.add("M");
listH.add("K");
listH.add(2, "I");
listH.add("AS");
listH.remove(1);
System.out.print(listH);
```

_____

I. What is output by the following code? The list in the question has already been declared and set to store the following elements: `[-5, 3, 2, 0, 5, 7]`

```
Iterator<Integer> it = list1.iterator();
System.out.print(it.next() + " ");
it.next();
it.remove();
System.out.print(it.next() + " ");
it.remove();
System.out.print(list1);
```

J. What is output by the following code? The list in the question has already been declared and set to store the following elements: `[12, 0, 19, 17, 0]`

```
Iterator<Integer> it = list2.iterator();
int jt = it.next();
it.remove();
jt += it.next();
list2.add(0);
jt += it.next();
it.remove();
System.out.print(jt + " " + list2);
```

For questions K through T, refer to the following classes.

```
public abstract class UTPerson {

      private int years;

      public UTPerson() { years = 18; }

      public UTPerson(int y) { years = y; }

      public abstract void advance();

      public void advance(int y) { years += y; }

      public int years() { return years; }

      public String toString() { return "y: " + years(); }
}

public interface Payable {
      public int pay();
      public String toString();
}

public class Student extends UTPerson {
      private int credits;

      public void advance() {
            credits += 15;
            advance(1);
      }

      public int years() { return credits / 10; }
}

public class Faculty extends UTPerson implements Payable {

      public Faculty() { super(30); }

      public int pay() { return years() * 2; }

      public void advance() { advance(2); }

      public String toString() { return "fac" + pay(); }
}

public class Lecturer extends Faculty {

      public String toString() { return "L:" + years(); }

      public boolean equals(Lecturer f) { return years() == f.years(); }
}
```

K. For each line of code write **valid** if the line will compile without error or **invalid** if it causes a compile error. (.5 points each)

```
Payable p1 = new Payable();          _____________________

Payable p2 = new Lecturer();         _____________________
```

L. For each line of code write **valid** if the line will compile without error or **invalid** if it causes a compile error. (.5 points each)

```
UTPerson p3 = new Faculty();         _____________________

Lecturer p4 = new Faculty();         _____________________
```

M. What is output by the following code? _____

```
Student s1 = new Student();
s1.advance(30);
s1.advance();
s1.advance();
System.out.print(s1);
```

N. What is output by the following code? _____

```
Faculty f1 = new Lecturer();
Faculty f2 = new Lecturer();
System.out.print(f1.equals(f2));
```

O. What is output by the following code? _____

```
Lecturer c1 = new Lecturer();
Lecturer c2 = new Lecturer();
c2.advance();
c1.advance();
System.out.print(c1.equals(c2) + " " + (c1 == c2));
```

P. What is output by the following code? _____

```
UTPerson[] ps = {new Faculty(), new Lecturer()};
for (UTPerson p : ps) {
     p.advance();
     System.out.print(p + " ");
}
```

Q. What is output by the following code? _____

```
Lecturer q2 = new Lecturer();
q2.advance();
q2.advance(10);
System.out.print(q2.pay() + " " + q2);
```

R. What is output by the following code? _____

```
Student r1 = new Student();
r1.advance();
r1.advance();
System.out.print ( ((UTPerson) r1).years() );
```

S. Will the following interface compile? If not, why not?

```
public interface Exempt extends Payable {

     private String tag = "EX";

     public int getBenefits();

     public boolean equals(Object obj);
}
```

T. Will the following class compile? If not, why not?

```
public class TA implements Payable {

     public int pay() { return 1000; }
}
```

---

2. The `GenericList` class (20 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 N elements of a native array. 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 be larger than the list it represents.

Complete an instance method for the `GenericList` class `removeAll`.

The method removes all elements in the list equal to a given value.

The relative order of the remaining elements in the list is unchanged.

The method returns the number of elements removed as a result of the method.

```
/*   pre: tgt != null
     post: Removes all elements from this list equal to tgt. The
relative order of the remaining elements in the list is unchanged.
     Returns the number of elements removed by this method.*/
public int removeAll(E tgt) {
```

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

```
[].removeAll(A) -> list unchanged, returns 0

[A, A].remove(A) -> list becomes [], returns 2

[A, A].remove(AA) -> list unchanged, returns 0

[AA, A, B, AA, B, AA, CC].remove(AA) -> list becomes [A, B, B, CC],
                                        returns 3
```

The `GenericList` class:

```
public class GenericList<E> {

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

You may not use any methods from the `GenericList` class unless you implement them yourself as a part of your solution. Do not use any other Java classes or methods except the `equals` method. Do not create any new arrays as in your solution.

The list does not store null elements.

Complete the method on the next page.

```
/*   pre: tgt != null
     post: Removes all elements from this list equal to tgt. The
relative order of the remaining elements in the list is unchanged.
     Returns the number of elements removed by this method. */
public int removeAll(E tgt) {
```

3. GenericList 2 (20 Points) This question uses the same `GenericList` as question 2.

Write an instance method for the GenericList class, `maxFrequency`.

The method accepts 3 parameters:

- Another GenericList. This reference shall not equal `null`.
- A target value. This reference shall not equal `null`.
- A starting index. This value shall be greater than or equal to 0.

The method returns a reference to the list (calling object or explicit parameter) that contains the most occurrences of the target element starting at the given index in the list.

If there is a tie for the most elements, return the list with the smaller size, unless neither list contains **any** elements equal to the target value starting at the starting index. In that case, return `null`.

Examples of calls to `maxFrequency(GenericList<E> other, E tgt, int start)` and the expected result. In these examples this lists contain Strings.

```
[].maxFrequency([A, BB, A, C], A, 0) -> returns reference to other

[].maxFrequency([A, BB, A, C], A, 1) -> returns reference to other

[].maxFrequency([A, BB, A, C], A, 2) -> returns reference to other

[].maxFrequency([A, BB, A, C], A, 3) -> returns null

[A, A].maxFrequency([A, BB, A, C], A, 0) -> returns reference to the
calling object

[A, A].maxFrequency([A, BB, A, C], A, 1) -> returns reference to the
calling object

[A, A].maxFrequency([A, BB, A, C], A, 2) -> returns reference to other

[A, A].maxFrequency([A, BB, A, C], A, 3) -> returns null

[A, A].maxFrequency([A, BB, A, C], A, 6) -> returns null

[A, A, B, CC, AA].maxFrequency([A, BB, A, C], X, 0) -> returns null
```

You may not use any methods from the `GenericList` class unless you implement them yourself as a part of your solution. Do not use any other Java classes or methods except the Object `equals` method.

Complete the method on the next page.

```
/*   pre: other != null, tgt != null, start >= 0
     post: per the problem description
public GenericList<E> maxFrequency(GenericList<E> other, E tgt,
                                                        int start) {
```

4. Maps (20 points) Write a method that calculates the *semantic descriptor vector* for a given word and a given text and stores the semantic descriptor in a map. Consider the following text, taken from the children's book *The Cat in the Hat*.

```
tell that cat in the hat your do not want to play
he should not be here
he should not be about
he should not be here when your mother is out
now now have no fear
```

Assume the text has been modified in the following ways:

- each sentence appears on a separate line
- all non-English letters have been removed except for a space before and after each word
- all letters have been converted to lower case
- a space has been added to the start and end of each sentence
- there are no blank lines

Given the text above the semantic descriptor for the word ***should*** is

[(not, 3), (be 3), (here, 2), (about, 1), (when, 1), (your, 1), (mother, 1), (is, 1), (out, 1), (he, 3)]

Note how the word ***should*** is in three sentences with the words he, not, and be. It is in two sentences with the word here and only one sentence with the words about, when, your, mother, is, and out. Note, if a word appears in a sentence with the target word multiple times, we count each occurrence separately. For example, if the target word is fear the sentence `now now have no fear` counts as two sentences.

The semantic descriptor for a word does not contain the word itself. In other words, even if there a sentence contained two or more instances of the word ***should*** the word ***should*** does not appear in the semantic descriptor for itself.

The order of the words in the resulting map is unimportant.

For this question you may create and use **one** `Map` object, either a `HashMap` or a `TreeMap`.

You may use the following methods from the `Map` interface.

put(K, V)	adds a mapping from the given key to the given value if key already present, replaces old value with given value
get(K)	returns the value mapped to the given key (null if none)
containsKey(Object)	returns true if this key is present in the map, false otherwise
Set<K> keySet()	returns a set with the keys of this map
V remove(K key)	remove the given key-value pair if present

You may create new `Scanner` objects as necessary. You may use the `hasNext(), hasNextLine(), next(),` and `nextLine()` methods from the `Scanner` class.

You may use the `contains(String str)`, `equals(Object obj)` methods from the `String` class and String concatenation.

You may obtain and use `Iterator` objects and the `hasNext(), next(), and remove()` methods from the `Iterator` class.
Do not use any other Java methods or classes.

```
/* pre: sc is connected to a data source as described in the problem.
word != null, word.length() >= 1, word only contains lower case
English letters. */
public Map<String, Integer> getSemanticDescriptor(Scanner sc, String word)
```

5. New Data Structures (20 points) A *Bag* or *Multiset* is an unordered collection of elements in which duplicates are allowed.

Because there is no notion of position for the client, the elements of a Bag may be stored in an array with gaps present in the internal array.

For example, the Bag (A, A, B, A, C) could be stored in the following array.
The lines represent references to the objects that are part of the Bag

```
[|, null, null, |, null, |, |, null, |, null]
 |              |        |  |        |
 A              A        B  A        C
```

Assume from a client's view, `null` elements are not allowed to be part of the Bag.
Therefore, a `null` in the array indicates an empty spot in the internal storage container.

Write an instance method that adds once instance of a given value to the `Bag`, **unless** the bag already contains the given value max (an `int` parameter) or more times.

Examples of calls to `boolean addIfFewerThan(E val, int max)`. The method returns `true` if `val` was added, `false` otherwise. The elements in the bag in these examples are Strings. **Note, if the value is added to the bag it is not necessarily at the end of bag as shown in these examples.**

```
(A, A, B, A, C).addIfFewerThan(A, 1) -> returns false, bag unaltered

(A, A, B, A, C).addIfFewerThan(A, 2) -> returns false, bag unaltered

(A, A, B, A, C).addIfFewerThan(A, 3) -> returns false, bag unaltered

(A, A, B, A, C).addIfFewerThan(A, 4) -> returns true,
                            bag becomes (A, A, B, A, C, A)

(A, A, B, A, C).addIfFewerThan(B, 1) -> returns false, bag unaltered

(A, A, B, A, C).addIfFewerThan(B, 2) -> returns true,
                            bag becomes (A, A, B, A, C, B)

(A, A, B, A, C).addIfFewerThan(QQ, 2) -> returns true,
                            bag becomes (A, A, B, A, C, QQ)
```

The `Bag` class for this question:

```
public class Bag<E> {

     private int size; // number of elements in Bag. In example size == 5
     private E[] con; // container for elements
```

You may not use any methods from the `Bag` class unless you implement them yourself as a part of your solution. Do not use any other Java classes or methods except the Object `equals` method and native arrays.

```
/*   pre: val != null, max >= 1
     post: per the problem description. */
public boolean addIfFewerThan(E val, int max) {
```