Points off	1	2	3	4	5	6	Total off	Net Score

CS 314 – Exam 1 – Spring 2018

Your Name_____

Your UTEID _____

Circle your TAs Name: Aish Anthony Chris Dayanny Hailey
Ivan Jacob Joseph Lucas Shelby

Instructions:

1. There are **6** questions on this test. 100 points available. Scores will be scaled to 170 points.
2. You have 2 hours to complete the test.
3. Place your final answers on this test. Not on 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, 15 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 other 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 should be the most restrictive correct Big O function. For example, Selection Sort has an average case Big O of $O(N^2)$, but per the formal definition of Big O it is correct to say Selection Sort also has a Big O of $O(N^3)$ or $O(N^4)$. 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 average case order of the following method?
Assume `Math.random()` is O(1). _____

```
public static double[] methodA(int n) {
     double[] result = new double[n];
     for (int i = 0; i < n; i++) {
          result[i] = Math.random();
          if (result[i] < 0.75) {
               for (int j = 0; j < i; j++) {
                    result[j] = result[i];
               }
          }
     }
     return result;
}
```

B. What is the best case order of the following method? N = `data.length`.

_____

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

C. What is output by the following code?

_____

```
ArrayList<String> listC = new ArrayList<>();
listC.add(0, "K");
listC.add("V");
listC.add(0, "J");
listC.add(1, listC.get(2));
listC.set(2, listC.get(0) + listC.size());
System.out.print(listC);
```

D. A method is $O(N^2)$. It takes the method 20 seconds to complete when N = 50,000. What is the expected time in seconds for the method to complete when N = 25,000?

_____

E. Consider the following timing data. Given the timing data what is the most likely order of the code being timed?

_____

N	Time
128,000	.01 seconds
256,000	.0106 seconds
512,000	.0112 seconds
1,024,000	.0118 seconds

F. A method is $O(N^4)$. It takes 1 second for the method to complete when N = 10,000. What is the expected time in seconds for the method to complete when N = 20,000?

_____

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

_____

```
public static void methodG(ArrayList<Integer> list, int tgt) {
     Iterator<Integer> it = list.iterator();
     while (it.hasNext()) {
          if (it.next() < tgt) {
               it.remove();
          }
     }
}
```

H. What is output by the following code? The `Map.toString` returns elements in the form `{key1=value1, key2=value2, ..., keyN=valueN}`.

```
Map<String, Integer> m = new TreeMap<>();
m.put("C", 5);
m.put("A", 5);
m.put("G", 12);
m.put("A", m.get("G"));
m.put("G", 10);
System.out.print(m);
```

_____

I. The following method takes 3 seconds to complete when `list.size()` = 50,000. What is the expected time in seconds for the method to complete when `list.size()` = 150,000?

_____

```
public static ArrayList<Integer> methodI(ArrayList<Integer> list) {
     ArrayList<Integer> result = new ArrayList<>();
     for (int i = 0; i < list.size(); i += 2) {
          result.add(list.get(i));
     }
     int last = 0;
     for (int i = 0; i < result.size(); i++) {
          last += result.get(i);
     }
     result.add(last);
     return result;
}
```

For questions J through O, refer to the classes defined on the opposite page.

J. What is output by the following code? _____

```
Movie m1 = new Movie();
System.out.print(m1.getIntro());
```

K. What is output by the following code? _____

```
TVShow t1 = new TVShow();
t1.putOnTV();
t1.putOnTV();
System.out.print(t1.getLength());
```

L. What is output by the following code? _____

```
Production p1 = new Movie();
System.out.print(p1.getLength() + " " + p1.hasCommercials());
```

M. What is output by the following code? _____

```
Production p2 = new Movie();
System.out.print(((TVShow) p2).getIntro());
```

N. What is output by the following code? _____

```
Production p3 = new Movie();
p3.putOnTV();
System.out.print(p3);
```

O. What is output by the following code? _____

```
Production p4 = new Production();
p4.putOnTV();
p4.putOnTV();
System.out.print(p4.getLength());
```

For questions J - O, consider the following classes.

```
public abstract class Production {

      private int length;

      public Production() { length = 20; }

      public Production(int len) { length = len; }

      public abstract boolean hasCommercials();

      public int getLength() { return length; }

      public void putOnTV() { length += 10; }

      public String toString(){ return "" + getLength() + hasCommercials(); }
}

public class Movie extends Production {

      private int intro;

      public Movie() { super(120);      }

      public boolean hasCommercials() { return false; }

      public int getIntro() { return intro; }

      public int getLength() { return 150; }
}

public class TVShow extends Production {

      public boolean hasCommercials() { return true; }

      public int getIntro() { return 5; }
}
```

2. The `GenericList` class (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 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 a method that returns the *mode* of the list. Recall the mode is the value that appears most often in the list.

```
/*   pre: size() > 0
     post: Return the mode of this list. If multiple elements are tied
     for the mode return the one closest to the beginning of the list.
     This list is not altered as a result of this method call.
*/
public E mode() {
```

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

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

[A, X, B, A, B, D, B].mode() -> returns B

[A].mode() -> returns A

[X, G, J, I, K] -> returns X
```

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.

The list does not store null elements.

Complete the method on the next page.

```
/*   pre: size() > 0
     post: Return the mode of this list. If multiple elements are tied
     for the mode return the one closest to the beginning of the list.
     This list is not altered as a result of this method call.
*/
public E mode() {
```

3. GenericList (17 points) This question uses the same `GenericList` class as described in question 2.

Create an instance method for the `GenericList` class `getRevCopyWithoutValue`.

The method accepts one parameter of type `E` named `val`. The method creates and returns a new `GenericList` with the same elements in the reverse order as the calling object, except there are no instances of the parameter `val` in the list returned by the method.

Consider the following examples. Are values shown are `String`s.

```
[].getRevCopyWithoutValue(A) -> returns []

[A, A, A, A, A, A].getRevCopyWithoutValue(A) -> returns []

[A, C, A, D, X, B, A, C, B].getRevCopyWithoutValue(A) -> returns
                                                    [B, C, B, X, D, C]

[C, D, X, B, C, B].getRevCopyWithoutValue(M) -> returns
                                                    [B, C, B, X, D, C]
```

The `GenericList` class:

```
public class GenericList<E> {

     private E[] con;
     private int size;

     public GenericList() { this(10); }

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

You may only use the constructors and fields from the `GenericList` class shown above.

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

You may call the `equals` method on objects.

The list does not store null elements.

Complete the method on the next page.

```
/*   pre: val != null
     post: Per the problem description. This list is not altered as a
     result of this method call.
*/
public GenericList<E> getRevCopyWithoutValue(E val) {
```

4. Math Matrix (17 Points) Create a method for the `MathMatrix` class that returns `true` if the calling object is an ***upper bidiagonal matrix***, `false` otherwise. An ***upper bidiagonal matrix*** is a square matrix with **nonzero** entries along the main diagonal (from upper left to lower right) and the diagonal **directly** above the main diagonal. All other entries in the matrix must be equal to **0**. The matrix must be square and have at least 2 columns to be upper bidiagonal.

For example:

```
m1:   4    0    0    0      not upper bidiagonal, values on diagonal above main diagonal not
      0    3    0    0      all non-zero values
      0    0   -5    0
      0    0    0    2

m2:   4    3    0    0      upper bidiagonal
      0    3    2    0
      0    0   -5    5
      0    0    0    2

m3:   4    0    0    0      not upper bidiagonal
      6    3    0    0
      0    7   -5    0
      0    0    3    2

m4:   4    6    0    0      not upper bidiagonal, 0 on the main diagonal
      0    3    7    0
      0    0    0    3
      0    0    0    2

m5:   4    5    0    0      not upper bidiagonal, diagonal above main not all non-zeros
      0    3    0    0
      0    0    5    3
      0    0    0    2

m6:   4    6    0    0      not upper bidiagonal, non-zero value not on main diagonal or diagonal
      0    3    7    0      above main diagonal
      0    0   -5    3
      0    2    0    2

m7:   4    6    0    0      not upper bidiagonal, not square
      0    3    7    0
      0    0   -5    3

m6:   4    6    upper bidiagonal
      0    3
```

Recall the `MathMatrix` class:

```
public class MathMatrix {

      private int[][] cells; // no extra capacity
```

Do not use any other Java classes besides `MathMatrix`. You may not use any other methods from the `MathMatrix` class unless you implement them yourself as a part of your answer to this question.

Complete the following instance method of the `MathMatrix` class.

```
/*   pre: none   post: return true if this is an upper bidiagonal
          matrix, false otherwise. This MathMatrix is not altered.*/
public boolean isUpperBidiagonal() {
```

5. Baby Names (17 points) Complete an instance method in the `Names` class that returns an `ArrayList<String>` of all the names that are in sync with a given name based on a maximum allowed difference.

A name is *in sync* with another if the differ by no more than some max value for **every** decade. Recall, a value of 0 indicates the name had a rank greater than 1000 for the decade. **For this question assume a 0 indicates a rank of 1001 when calculating the difference between ranks.** Consider the following example:

```
Olivia       900 311 290 307 270 355 504 324 193  47 16
Isaac          0 197 203 260 269 322 383 211 154 102 53
Difference:  101 114  87  47   1  33 121 113  39  55 37
```

The absolute value of the difference between the ranks of Olivia and Isaac never exceeds 121. Therefore, if the max difference allowed is 121 or more Olivia and Isaac are in sync. If the max allowed difference is 120 or less, Olivia and Isaac are not in sync.

With a given max allowed difference of 150 Olivia is in sync with these names: Andres, Claire, Dominic, Isaac, and Julian. Note, a name is not in sync with itself.

The `Names` class you will use on this question is as follows:

```
public class Names {
     // the NameRecords in this Names object.
     // All NameRecords have the same number of decades.
     private ArrayList<NameRecord> names;

     // get a NameRecord for the given name. Returns null if not present
     public NameRecord getRecord(String name)
}
```

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 in `for-each` loops.

You may use the `equals` method on `String`s and the `Math.abs` method. **Do not use any other Java classes or methods besides those listed above.**

```
/* pre: name != null, there is a NameRecord with name in this Names
     object, maxDiff >= 0
  post: return an ArrayList<String> of names in sync with name.
     Names is not altered as a result of this method call. */
public ArrayList<String> getNamesInSync(String name, int maxDiff) {
```

6. Other Data Structures (17 points) In class we implemented lists that store every value in the list explicitly. However, what if most of the elements of the list equal the same value? Consider the following list:

```
 0  1  2  3  4  5  6  7  8  9  10   11 12 13 14 15 16  // position
[A, B, A, A, A, A, A, A, A, A, AAA, A, B, A, A, C, A] // element
```

This is the abstract view of the list. Storing all those A's seems like a waste of space. In a ***sparse list***, only the elements not equal to the default value are explicitly stored. The default value is set when the list is created and does not change for a given list. Internally we use a native array as our storage container so we also store the position of each element, because the position in the array does not necessarily equal the position in the list.

Consider the following internal representation of the list shown above. Each element in the array is a `ListElem` object that stores one element of data and the position of that element in the list. `(position, non-default element)`

The elements not equal to the default element, are stored in the array in ascending order based on their position in the list. This is a more concrete view of how we represent this list.

```
     0          1         2         3      4     5  index in array
[(1, B), (10, AAA), (12, B), (15, C), null, null]
elements stored = 4, sizeOfList = 17
All elements not stored explicitly equal A for this list.
```

Complete the `ArrrayList<E> getExplicitList()` method for a `SparseList` class. The method returns a non-sparse version of the calling object as an `ArrayList`.
For example, given the sparse list above the method would return an `ArrayList` with elements equal to the example at the top of the page.

Here is the `ListElem` class:

```
public class ListElem<E> {

     public E getData() // return data of this element
     public int getPosition() // return position of this element
```

The properties of the `SparseList` class are:

- the internal storage container is a native array of `ListElem` objects
- there may be extra capacity in the native array
- only elements not equal to the default element are stored explicitly
- the non-default elements are stored at the beginning of the array in ascending order based on their position in the list
- the size of the list, the number of elements stored explicitly in the array, and the default value are stored in separate instance variables
- any elements in the array that are not referring to active elements of the list are set to `null`
- the default list value never equals `null`

```
public class SparseList<E> {

     private ListElem<E>[] values;
     private int sizeOfList;

     // Elements not stored explicitly in values equal defaultValue.
     // defaultValue never equals null.
     private final E defaultValue;

     // Number of elements stored explicitly in values.
     // The elements are stored at the beginning of the array.
     // This value could be 0 even if sizeOfList > 0 indicating
     // every element in the list is the default value.
     private int elementsStored;
```

You may use the following methods and constructors from the `ArrayList` class:

```
public ArrayList() // create empty list with capacity 10
public boolean add(int pos, E val) // insert val at position
public boolean add(E val) // add val to end of list
```

Complete the **`getExplicitList()`** instance method for the `SpasesList` class on the next page.

QUESTION (3 points of 20): What is the danger in adding multiple references to the `defaultValue` to the resulting `ArrayList` returned by the `getExplicitList` method? You are *not* expected to avoid this danger in your solution to `getExplicitList`.

```
/* pre: none   post: per the problem description. This object is not
altered as a result of this method. */
public ArrayList<E> getExplicitList() {
```