Midterm 1: CompSci 201
Form B

Profs. Astrachan and Nemecek

October 7, 2025

Name: _____

NetID: _____

In submitting this test, I affirm that I have followed the Duke Community Standard.

Community standard acknowledgement (signature) _____

Be sure to write your name on *all three* of your exam packet, your answer sheet, and your notes sheets (if you used any). You will have to turn all of these in at the end of the exam.

Some questions have two only two answers to choose from, some have three, some have four. Make sure you bubble answers appropriately.

This Exam is Form B, please make sure your answer sheet is marked accordingly.

PROBLEM 1:

Consider the code to copy values from an array into an `ArrayList` shown below.

```
26  public ArrayList<String> copy(String[] a){
27      ArrayList<String> list = new ArrayList<>();
28      for(String s : a){
29          list.add(s);
30      }
31      return list;
32  }
```

Which **one of the following** is true about the code on line 29 when parameter `a` has a large number of values?

A. Every call to `list.add(s)` is $O(1)$.

B. There is at least one call that takes $O(n)$ time, but the total time for all calls is also $O(n)$.

C. Every call to `list.add(s)` is $O(n)$.

PROBLEM 2:

Consider the code to copy values from an array into a `HashSet` shown below (assume hashing works as we've discussed in class).

```
34  public Set<String> copySet(String[] a){
35      Set<String> set = new HashSet<>();
36      for(String s : a){
37          set.add(s);
38      }
39      return set;
40  }
```

Which **one of the following** is true about the code on line 37 when parameter `a` has a large number of values?

A. Every call to `set.add(s)` is $O(1)$.

B. There is at least one call that takes $O(n)$ time, but the total time for all calls is also $O(n)$.

C. Every call to `set.add(s)` is $O(n)$.

PROBLEM 3:

When the statements below are executed, what is printed?

```
String[] a = {"the", "quick", "fox", "jumped"};
String[] b = a;
a[1] = b[3];
System.out.println(a[1] + ", " + b[1]);
```

A. jumped, quick

B. jumped, jumped

C. fox, the

D. fox, fox

Asymptotic Runtime

The next two problems concern the call `number(n)` for the method `number` shown below. Note that `number(100) = 10100`.

```
public int number(int n){
   int total = 0;
   for(int k=1; k <= n; k++){
       total += k;
   }
   for(int k=1; k <= n; k++){
       total += k;
   }
   return total;
}
```

PROBLEM 4:

Using O-notation, what is the *runtime* of the call `number(n)`?

A. $O(n)$

B. $O(n \log(n))$

C. $O(n^2)$

D. $O(n^3)$

PROBLEM 5:

Using O-notation, what is the *value returned* by the call `number(n)`?

A. $O(n)$

B. $O(n \log(n))$

C. $O(n^2)$

D. $O(n^3)$

The next two problems concern the call `figure(n)` for the method `figure` shown below. Note that `figure(1024) == 1046 == 22+1024`.

```
public int figure(int n) {
    int total = 0;
    for(int k=1; k <= n; k *= 2) {
        total += 2;
    }
    for(int j=0; j < n; j++) {
        total += 1;
    }
    return total;
}
```

PROBLEM 6:

Using O-notation, what is the *runtime* of the call `figure(n)`?

A. $O(\log(n))$

B. $O(n)$

C. $O(n\log(n))$

D. $O(n^2)$

PROBLEM 7:

Using O-notation, what is the *value returned* by the call `figure(n)`?

A. Asymptotically less than the runtime of `figure(n)`.

B. Asymptotically the same as the runtime of `figure(n)`.

C. Asymptotically greater than the runtime of `figure(n)`.

The next two problems concern the call `sumNum(n)` for the method `sumNum` shown below.

```
public static int sumNum(int num) {
    int sum = 0;
    for(int i=num; i < num*num; i += num) {
        sum += num;
    }
    return sum;
}
```

PROBLEM 8:

Using O-notation, what is the *runtime* of the call `sumNum(n)`?

A. $O(\log(n))$

B. $O(n)$

C. $O(n\log(n))$

D. $O(n^2)$

E. $O(n^3)$

PROBLEM 9:

Using O-notation, what is the *value returned* by the call `sumNum(n)`?

A. Asymptotically less than the runtime of `sumNum(n)`.

B. Asymptotically the same as the runtime of `sumNum(n)`.

C. Asymptotically greater than the runtime of `sumNum(n)`.

The next two problems concern the method `create` shown below. Note that `create("duke",5).length()` is 20.

In class we discussed that the *runtime* of `create("duke",n)` is $O(n^2)$ because Strings are immutable.

```
public String create(String base, int n) {
    String ret = "";
    for(int k=0; k < n; k++){
        ret += base;
    }
    return ret;
}
```

PROBLEM 10:

Using O-notation, what is the *length* of the the String *returned* by the call `create("duke",n)`?

A. $O(n)$

B. $O(n^2)$

C. $O(n^3)$

D. $O(n^4)$

PROBLEM 11:

Using O-notation, what is the *runtime* of executing `create(create("duke",n),n)`? Note that `create(create("duke",2),2)` returns a String of length 16.

A. $O(n)$

B. $O(n^2)$

C. $O(n^3)$

D. $O(n^4)$

The next seven problems concern concepts and code from P0: Person201.

PROBLEM 12:

The code in `averageDistance` below returns the average distance between parameter `center` and all the other `Person201` objects referenced via the array parameter `data`. The `Person201` object `center` is referenced/stored somewhere in parameter `data`.

```
94  public static double averageDistance(Person201[] data, Person201 center){
95      double total = 0.0;
96      for(Person201 p : data){
97          double dist = Person201Utilities.distance(p, center);
98          total += dist;
99      }
100     return total/(data.length-1);
101 }
```

A student asks if adding the code below between lines 96 and 97 would make the code better.

```
if (p == center) continue;
```

Which of the following is the best response to this student's question?

A. The return value of the method will be the same with (or without) the `if ... continue` statement, so it is **not** needed and is **not** better with the addition.

B. Comparing a `Person201` object to itself should be avoided, but the test should use `if (p.equals(center))` rather than using `==` since using `.equals` is faster and more correct.

C. It is advisable to add such a statement, but the call to `distance` should be `Person201Utilities.distance(center,p)` rather than what is shown.

PROBLEM 13:

If `data` has N elements, which **one of the following** is true about the code below?

```
double total = 0.0;
for(int k=0; k < data.length; k++){
    total += averageDistance(data,data[k]);
}
// final value of total here
```

A. The calls to `averageDistance` have the same big-Oh runtime complexity for all values of `k`.

B. The calls to `averageDistance` return the same value for each value of `k`.

C. If for every array `data` such that `data.length <= 1000` and each `Person201` object in the array represents a valid location on Earth, the final value of `total` after the loop shown will be less than 5000.

D. For every array `data` such that `data.length <= 1000` and each `Person201` object in the array represents a valid location on Earth, the final value of `total` after the loop shown will be more than 100.

PROBLEM 14:

The method `centroid` finds the element of parameter `data` that has the smallest average distance to all other elements of `data`.

```
102  public static void centroid(Person201[] data){
103      Person201 best = null;
104      double min = Double.MAX_VALUE;
105      for(Person201 p : data){
106          double avg = averageDistance(data, p);
107          if (avg < min) {
108              min = avg;
109              best = p;
110          }
111      }
112      System.out.printf("most central %1.2f: %s\n",min,best);
113  }
```

The code in `centroid` shows a call to `averageDistance` shown in the previous problem. If there are N objects in parameter `data`, what is the overall runtime of calling `centroid(data)` *including calls* made to `averageDistance`?

A. $O(N)$

B. $O(N^2)$

C. $O(N^3)$

D. It cannot be determined without knowing the range of latitudes and longitudes of objects referenced in `data`.

PROBLEM 15:

If line 107 is changed to `if (avg > min)` and no other changes are made to the code, what is printed as the value of `best` on line 112?

A. The value of `data[0].toString()`.

B. The value of `data[data.length-1].toString()`.

C. The String `null`.

D. It cannot be determined from the information given.

PROBLEM 16:

A different measure of centrality is shown in method `centroid2` below. This returns the `Person201` object that has the most "nearby" `Person201` objects within the array `data`, where "nearby" is within the distance of parameter `epsilon`. For example, for the three calls below:

```
Person201[] pa = readData();
centroid2(pa,1000);
centroid2(pa,2000);
centroid2(pa,3000);
```

the output might be (some parts of each output line not shown):

```
best for epsilon = 1000.00 is (037.36N,079.28W) Angie, ... with 91
best for epsilon = 2000.00 is (028.61N,080.60W) Elmer, ... with 110
best for epsilon = 3000.00 is (032.78N,096.80E) Chris, ... with 132
```

```
115  public static void centroid2(Person201[] data, double epsilon){
116      Person201 best = null;
117      int max = 0;
118      for(int j=0; j < data.length; j++){
119          Person201 p = data[j]; // treat p as "possible center"
120          int count = 0;
121          for(int k=_____; k < data.length; k++){
122              Person201 q = data[k];
123              if (Person201Utilities.distance(p,q) < epsilon){
124                  count += 1;
125              }
126          }
127          if (count > max) {
128              max = count;
129              best = p;
130          }
131      }
132      System.out.printf("best for epsilon = %1.2f is %s with %d\n",
133                        epsilon,best,max);
134  }
```

The `Person201 p` object on line 119 is considered as a possible "center", and then the loop on lines 121-126 counts how many elements of `data` are within `epsilon` of `p`.

What value should fill in the blank line to initialize `k` on line 121 so that the code works as intended?

A. j

B. j+1

C. 0

D. 1

The following two problems are true or false questions.

PROBLEM 17:

The runtime complexity of the code shown does not depend on the answer to the previous problem.

A. True, the runtime has the same big-Oh runtime for each of those values.

B. False, the big-Oh runtime for the choice 0 or 1 is not the same as the runtime for the choice `j` or `j+1`.

PROBLEM 18:

For any `Person201[] data` array, there **is a value** for `epsilon` such that the output generated by the call `centroid2(data,epsilon)` as shown below will print `data.length` for `YYYY`.

```
best for epsilon = ???? is , ... with YYYY
```

A. True, there is always some value for `epsilon` that results in the output shown where `YYYY` has the value `data.length`.

B. False, there is not always a value that generates `YYYY` having the value `data.length`.

Hashing Stuff

For each statement, choose between True, False, or Cannot be determined from the information given. If you choose *cannot be determined*, that means for some values of variables `s` and `t` the statement could be true, but for other values it could be false. If you choose true, then the statement is true for any/all values of `s` and `t`, and similarly if you choose false, the statement is false for any/all values.

PROBLEM 19:

If two string variables `s` and `t` have non-null values, but then the assignment `s = t` is made, then what is the value of `s.hashCode() == t.hashCode()`?

A. True

B. False

C. Cannot be determined from the information given

PROBLEM 20:

If for two string variables `s` and `t` we know `s == t` is true, what is the the value of `s.equals(t)`?

A. True

B. False

C. Cannot be determined from the information given

PROBLEM 21:

If for two string variables `s` and `t` we know `s.equals(t)` is **false**, what is the value of `s.hashCode() == t.hashCode()`?

A. True

B. False

C. Cannot be determined from the information given

The next five problems concern concepts and code from P1:Markov.

PROBLEM 22:

A correct implementation of method `getFollows` from `SimpleMarkovModel` is shown below. If the model is trained on a file with T strings, what is the runtime of `getFollows`? (Recall that `myWordSequence` contains all the strings read during training.)

```
24  @Override
25  protected List<String> getFollows(List<String> context) {
26      List<String> follows = new ArrayList<>();
27      for(int start=0; start <= myWordSequence.size()-myModelSize; start += 1){
28          if (myWordSequence.subList(start, start+myModelSize).equals(context)){
29              follows.add(myWordSequence.get(start+myModelSize));
30          }
31      }
32      return follows;
33  }
```

A. $O(1)$

B. $O(T)$

C. $O(T^2)$

PROBLEM 23:

A correct implementation of method `getFollows` from `HashMarkovModel` is shown below. If the model is trained on a file with T strings, what is the runtime of `getFollows`?

```
29  @Override
30  protected List<String> getFollows(List<String> context) {
31      if (! myMap.containsKey(context)) {
32          return Collections.emptyList();
33      }
34      List<String> ret =  myMap.get(context);
35      return ret;
36  }
```

A. $O(1)$

B. $O(T)$

C. $O(T^2)$

PROBLEM 24:

The implementions of the method `getFollows` in the previous two problems are each prefaced with `@Override`. Which **one of the following** is true? Note that `getFollows` is called from `BaseMarkovModel.randomNextString`.

A. `@Override` is required since there is an implementation in `BaseMarkovModel`.

B. `@Override` is **not** required, but is considered good programming practice.

PROBLEM 25:

Which **one of the following is FALSE** regarding the implementations of Markov training and random text generation using `SimpleMarkovModel` and `HashMarkovModel`?

A. For training texts with T Strings, reading all the texts and initializing instance variables takes $O(T)$ time for both models.

B. Generating N random strings is much faster with `HashMarkovModel` than with `SimpleMarkovModel` when trained on the same texts with the same Markov order.

C. Both classes extend `BaseMarkovModel` and can access protected instance variables and methods from that class.

D. If `model.setSeed(123456)` is used on each model (same order, same training text) immediately followed by generating N random strings, the outputs for `Simple` and `Hash` models could be different from each other.

PROBLEM 26:

Consider an order M model, where each context is a `List<String>` whose size is M, after being trained on a text with T strings. A new method `getUniqueContextSize` returns the number of different/unique contexts in the trained model.

Note that `SimpleMarkovModel` has one instance variable for accessing contexts:

```
List<String> myWordSequence
```

whereas `HashMarkovModel` has that and the instance variable:

```
Map<List<String>,List<String>> myMap
```

Which **one of the following is FALSE**?

A. The implementation of `getUniqueContextSize` in `SimpleMarkovModel` requires at least $O(T)$ time.

B. The implementation of `getUniqueContextSize` in `HashMarkovModel` can be done in $O(1)$ time.

C. The number of unique contexts when trained on T strings increases as the order of the model M increases for any training text.

The next six problems relate to a class `BirdInfo` and a data file named `birds.csv` with 51 lines. The file includes the lines shown below, with each line containing the name of a US State, followed by a comma, followed by the state bird of that state, e.g., the state bird of Michigan is the American robin. (There is a line for Washington, D.C.)

```
Connecticut,American robin
Michigan,American robin
Maryland,Baltimore oriole
Louisiana,Brown pelican
Arizona,Cactus wren
Wyoming,Western meadowlark
Oregon,Western meadowlark
```

PROBLEM 27:

The method `readBirds` in class `BirdInfo` shown below correctly reads this file. When the method has executed, the value of `myBirds.get("Michigan")` is `"American robin"`.

```
7  public BirdInfo(){
8      myBirds = new HashMap<>();
9  }
10
11 public void readBirds(String fileName) throws IOException{
12     Scanner s = new Scanner(new File(fileName));
13     while (s.hasNextLine()){
14         String line = s.nextLine();
15         String[] data = line.split(",");
16         String state = data[0];
17         String bird = data[1];
18         myBirds.put(state,bird);
19     }
```

Which **one of the following** is true?

A. If lines 16 and 17 are interchanged (so that line 16 is `String bird = data[1]`) the code will create an incorrect map.

B. The map `myBirds` is an instance variable that has been initialized when `readBirds` executes.

C. If line 18 is replaced with `myBirds.put(data[0],data[1])`, the code will create an incorrect map.

The next two problems concern method `reverse` described below.

The method `reverse` is intended to return a map in which each key is the name of a bird and the corresponding value is an `ArrayList` of those states for which the key is the state bird. For example, the map returned is partially shown below, with bird keys on the left and the list of states with that state bird in brackets on the right.

```
Baltimore oriole:   [Maryland]
Eastern bluebird:   [Missouri, New York]
American robin:     [Connecticut, Wisconsin, Michigan]
Chickadee:          [Maine]
```

```
22  public Map<String,ArrayList<String>> reverse(){
23      Map<String,ArrayList<String>> map = new HashMap<>();
24      for(String state : myBirds.keySet()){
25          String bird =       expression 1
26          if (! map.containsKey(bird)){
27              map.put(bird,new ArrayList<>());
28          }
29          statement 1
30      }
31      return map;
32  }
```

PROBLEM 28:

What is the value of *expression 1* on line 25 so that the code works as intended?

Bubble A for this question on the front of the answer sheet and fill-in-the blank area for this question on the back of the bubble answer sheet. Your answer should reference either `map` or `myBirds`.

PROBLEM 29:

What is the value of *statement 1* on line 29 so that the code works as intended?

Bubble A for this question on the front of the answer sheet and fill-in-the blank area for this question on the back of the bubble answer sheet. Your answer should reference either `map` or `myBirds`.

PROBLEM 30:

When `reverse` executes correctly, it is possible that for the `map` returned there is *some* `String bird` that was included in `birds.csv` such that `map.get(bird).size() == 0`.

A. True, for the code shown that is possible

B. False, for the code shown that is not possible

PROBLEM 31:

If `revmap` is returned by the call `reverse` shown in the previous problems, what value is printed by the code below (assuming all code is run after reading the 51 lines in `birds.csv` described at the beginning of these problems).

```
int total = 0;
for(ArrayList<String> list : revmap.values()){
    total += list.size();
}
System.out.printf("all sizes: %d\n",total);
```

A. A number less than 51

B. The number 51

C. A number greater than 51

D. It cannot be determined without knowing the entire contents of `birds.csv`.

PROBLEM 32:

Given the method `reverse` shown previously, the call `mostStates(reverse())` returns `"Northern cardinal"` since that bird is the state bird for seven states: `[North Carolina, Indiana, Kentucky, Illinois, Virginia, West Virginia, Ohio]`.

```
34  public String mostStates(Map<String,ArrayList<String>> map){
35      String most = "";
36      int max = 0;
37      for(String bird : map.keySet()){
38          if (      expression       > max) {
39              most = bird;
40              max =       expression
41          }
42      }
43      //System.out.println(map.get(most));
44      return most;
45  }
```

The value for **expression** on lines 38 and 40 should be same for `mostStates` to work correctly. What is that expression?

Bubble A for this question on the front of the answer sheet and fill-in-the blank area for this question on the back of the bubble answer sheet. Your answer should reference parameter `map`.