A. Give brief examples where one would be more appropriate that the other.

For example

- Data structures: HashSet vs. TreeSet
  - HashSet: large number of examples where you’ll be adding or searching for enough elements such that the average-case \( O(1) \) operations for HashSet will yield better performance than the \( O(\log n) \) operations for TreeSet
  - TreeSet: You want the items in sorted order

I. Primitive types and how they can be used: int vs. double

II. an ArrayList (e.g. ArrayList<String> list) vs. an array (e.g. String[] a)

III. Analysis: empirical vs. mathematical

IV. Types of methods: static vs. dynamic (i.e., not static - no modifier)

B. For each of the following object-oriented programming terms, summarize the distinction between the two terms. Your answer should be brief.

I. Comparing objects via == vs. .equals

II. Classes that implement Comparable vs. Comparator

III. class vs. object

IV. the constructor vs. any other class method

C. Pat is getting a code review for the WordNgram class in the Markov assignment. Pat claims that the implementation below passes all JUnit tests for methods equals and hashCode – Pat is confident these methods are correct. Chris says the performance of the algorithms using this implementation will be poor, especially when WordNgram objects are inserted into and retrieved from a HashMap or TreeMap.

```java
public class WordNgram implements Comparable<WordNgram>{
    private String[] myWords;

    public boolean equals(Object o){
```
if (this == o) // point to the same Object
return true;
if (o == null || getClass() != o.getClass())
return false;
WordNgram wg = (WordNgram) o;
for(int k=0; k < myWords.length; k++) {
    if (myWords[k] != wg.myWords[k]) return false;
}
return true;

public int hashCode(){
    return myWords.length;
}

public int compareTo(WordNgram wg) {
    return myWords.length() - wg.myWords.length();
}

    // some methods not shown

Explain why Chris and Pat are both right: give a brief explanation for why the code is correct and why the performance is poor for both HashMaps and TreeMaps.

D. In the Markov assignment, the WordNgram class includes a equals method similar to the one below.

public class WordNgram {
    private String[] myWords; // other methods omitted
    public boolean equals( Object o) {
        Object other = (WordNgram) o;
        for(int k=0; k < myWords.length; k++) {
            if (!myWords[k].equals(other.myWords[k]))
                return false;
        }
        return true;
    }
}

Why is the parameter, o, of type Object and why is it cast to a WordNgram?

E. Consider the following program.

public class Pattern {
    public static void print(int n) {
        for (int i = -n; i <= n; i++) {
            for (int j = -n; j <= n; j++) {
                if (i == j)
                    System.out.print("A ");
                else if (i == -j)
                    System.out.print("B ");
                else
                    System.out.print(". ");
        }
    }
}
System.out.println();
}
}

public static void main(String[] args) {
  // Call print method with 1 as the argument
}

I. Add a line to the main method above to call the print method with 1 as the argument.
II. What is the result of calling print above?
III. What is the result of calling print with 2 as the argument?

F. (4) Suppose that b[] is an array of 100 elements, with all entries initialized to 0, and a[] is an array of N elements, each of which is an integer between 0 and 99. What is the effect of the following loop? (select all that apply)

for (int j = 0; j < N; j++)
  b[a[j]]++;

I. Sets b[0] to 0, b[1] to 1, b[2] to 2, etc.
II. Sets b[0] to the number of 0s in a[], b[1] to the number of 1s in a[], etc.
III. Sets b[0] to a[0], b[1] to a[0] + a[1], b[2] to a[0] + a[1] + a[2], etc.
IV. Sets all entries of b[] to 1.
V. Out-of-bounds array access (i.e. ArrayIndexOutOfBoundsException.
VI. None of the above.
G. What is printed as a result of the following code excerpts?

I. ```java
ArrayList<String> names = new ArrayList<String>();
TreeSet<String> namesSet = new TreeSet<String>();
names.add("Mary");
names.add("Jack");
names.add("Mary");
names.add("Jack");
namesSet.addAll(names);
System.out.println(names);
System.out.println(namesSet);
``` 

II. ```java
Map<Integer,String> mappings = new TreeMap<Integer,String>();
mappings.put(7, "D");
mappings.put(3, "C");
mappings.put(6, "B");
mappings.put(6, "A");
for (Integer num: mappings.keySet())
{
    System.out.println(num + " " + mappings.get(num));
}
``` 

H. Provide big-Oh expressions for both the running time and the value returned for `stuff(n)` below. Justify these.

```java
public int stuff(int n){
    int sum = 0;
    for(int k=0; k < n; k++){
        sum += n;
    }
    for(int k=0; k < n; k++){
        sum += 2;
    }
    for(int k=0; k < n; k++){
        sum += 3;
    }
    return sum;
}
``` 

PROBLEM 2: (Close to Home (10 points))

In this problem, you will write a method `closestToOrigin` that given arrays of x-coordinates and y-coordinates corresponding to points, returns the point that is the minimum distance away from the origin (0,0).

For example, given the arrays

`closestToOrigin({1, 0, 6, 3, 5}, {10, -12, 4, 4, 5})`

should return the Point (3,4). The class `Point` is defined as follows:

```java
public class Point
```
{  
    public int x;
    public int y;

    public Point(int x, int y){
      this.x = x;
      this.y = y;
    }  

    public double distanceFrom(Point p){
      return Math.sqrt( (x-p.x)*(x-p.x) + (y-p.y)*(y-p.y) );
    }  
}

Complete closestToOrigin below.

    public static Point closestToOrigin (int[] xVals, int[] yVals)
    {

PROBLEM 3: (Comparisons)
The ABC organization has many chapters in every state of the United States. In order to keep track of all the member information an ABCChapter class has been created. An ABCChapter has three kinds of information, an array of all its member names, the state and the region of the state. We would like to be able to compare ABCChapters and thus to implement Comparable. Some of its methods are shown below and on the next page.

Assume that each member is a string with ?Lastname Firstname? and that the members in each chapter are stored in myMembers in alphabetical order. For example one ABC Chapter with four members might be:

State = "NC"
Region = "Eastern"
MemberList = "Astrachan Owen", "Duvall Robert" "Forbes Jeff" "Rodger Susan"

Below is the class.

public class ABCChapter implements Comparable<ABCChapter>{
  private String[] myMembers;
  private String myState;
  private String myRegion;

  public ABCChapter(String[] list, String state, String region) {
    myMembers = new String[list.length];
    System.arraycopy(list, 0, myMembers, 0, list.length);
    myState = state;
    myRegion = region;
  }

  /**
   * Return value that meets criteria of compareTo conventions.
   * @param chap is the ABCChapter to which this is compared
   * @return appropriate value less than zero, zero, or greater than zero
   */
  public int compareTo(ABCChapter chap) { // TODO implement this method
    }
/**
 * Return true if this ABCChapter is the same as the parameter
 * @param o is the ABCChapter to which this one is compared
 * @return true if o is equal to this ABCChapter
 */
public boolean equals(Object o){
    ABCChapter chap = (ABCChapter) o;
    // TODO return correct value
}
/**
 * Return a good value for this ABCChapter to be used in hashing.
 */
public int hashCode(){ // TODO return a hash value

A. Implement the equals method for the ABCChapter class. Two ABCChapters are equal if they have the same state, the same region, and all the members are the same. Assume the members are stored in alphabetical order in myMembers.

public boolean equals(Object o){
    ABCChapter chap = (ABCChapter) o;

B. Implement the compareTo method for the ABCChapter class. An ABCChapter is less than another ABCChapter if they have fewer members. If both chapters have the same number of members, then a chapter is less than another chapter if its state comes before it in alphabetical order. If the states are the same then a chapter is less than another chapter if its region comes before the other region in alphabetical order. If both the regions and the state are the same, then one chapter is less than another chapter if the list of member names comes before the list of the other chapter member names in alphabetical order.

public int compareTo(ABCChapter chap) {

C. Implement the hashCode method for the ABCChapter class and explain why you think it is a good hashCode function.

public int hashCode(){

PROBLEM 4 : (Like MemberCheck)
A health club chain allows its members to visit any of its many health club locations an unlimited number of times per day. The only constraining rule is, a customer can only visit one health club location per day, even though he or she may return to that location an unlimited number of times for the rest of that day. Although the honor system has always worked quite well, the club wants to run some tests to see how many people really follow the rules. You are to write a method that takes the entrance log files from all of the different clubs (all logging the same day) and return a sorted list of the people who are not honest and went to more than one health club location in the same day.
The entrances for each club are listed on a line. For example in the following file, there are three clubs.

JOHN JOHN FRED PEG
PEG GEORGE
GEORGE DAVID GEORGE

In Club 0, John visited twice, while Fred and Peg visited once. Peg and George each visited Club 1 once. George also visited Club 2 twice, and David visited Club 2 once.
A. Complete the method, `attendeeList` below so that it returns a List of Sets corresponding to the members who visited each club.

For the above file, the three element list below should be returned.

```
{[FRED, JOHN, PEG], [GEORGE, PEG], [DAVID, GEORGE]}
```

```java
List<Set<String>> attendeeList(Scanner in) {
}
```

B. Write a method, `whosDishonest` that takes the entrance log files data structure from `attendeeList` the previous part and return a sorted list of the people who are not honest and went to more than one health club location in the same day.

For the file on the previous page, the two element list `{GEORGE, PEG}` should be returned.

```java
List<String> whosDishonest(List<Set<String>> attendees) {
}
```

C. Briefly describe the big-Oh of the following call to your methods in terms of \( n \), the number of clubs, and \( m \), the number of people who visit each club per day. Briefly justify your answer

```java
Scanner in = getScanner(); // open log file
List<String> list = whosDishonest(attendeeList(in)); // Give big-Oh for this line
```

PROBLEM 5: (Sets)

You are given the data from Dook University of the clubs and members of each club in the following format. The data for each club is on a line. Each line has the name of all the members in the club, with members separated by a colon.

The sample data below shows the members of three clubs.

```
Jeff Forbes:Hillary Rodham Clinton:Mary Lou Retton
Susan Rodger:Oprah Winfrey:Cay Horstmann:Mary Lou Retton:Owen Astrachan
Owen Astrachan:Oprah Winfrey:Mary Lou Retton
```

A. Complete the Clubs method `allMembers` that given a `Scanner` as an argument that is initialized to read from a valid data file, returns an ArrayList of all the people from Dook University in a club.

For example, using the datafile given above, the `ArrayList` would contain: Jeff Forbes, Hillary Rodham Clinton, Mary Lou Retton, Susan Rodger, Oprah Winfrey, Cay Horstmann, and Owen Astrachan.

You may write and use helper functions if you find it helpful.

```java
public class Clubs {
    public ArrayList<String> allMembers(Scanner in) {
```

B. Describe how you would change `allMembers` to return an `ArrayList` of all the people who are in more than one club.

In the previous example, you would return the list containing: Mary Lou Retton, Owen Astrachan, and Oprah Winfrey.

You should be specific as possible in describing your changes. For example, you could specify exactly the lines of code that would change and what the new lines would be.
PROBLEM 6: (Map)

```java
public class FreqCompare implements Comparator<String> {
    private HashMap<String, Integer> myMap;

    public FreqCompare(String[] a) {
        // To be filled in..
    }

    public int compare(String a, String b) {
        int afreq = myMap.get(a);
        int bfreq = myMap.get(b);
        int diff = bfreq - afreq;
        if (diff != 0)
            return diff;
        return a.compareTo(b);
    }
}
```

A. Complete the constructor for FreqCompare below, so that the map is correctly filled in. That is, myMap should map from entries in the original array to the number of times they occur.

```java
public FreqCompare(String[] a) {
}
```

B. Describe how the ordering done by the Comparator will change if the line

```java
int diff = bfreq-afreq;
```

is changed to the following. Be brief and precise, not thorough.

```java
int diff = afreq-bfreq;
```

That is, given the following code:

```java
String a[] = // OMITTED
Arrays.sort(a, new FreqCompare(a));
```

how would the contents of a change.

PROBLEM 7: (How many fit in the pool?)

Write a method uniqueNames that has one Scanner parameter. This method reads names from a file and returns an array of the unique last names in alphabetical order.

The data is a list of names, one name per line. The last name is the last word on the line. There is exactly one space between words on a line. Here is a sample data file:

```
John Kevin Smith
Greg Smith
Fred El Mira Gumption
Chantal LaFruit
Sarah Ann Gumption
```
For this data file bound to the Scanner input, the call uniqueNames(input) would return the array
{"Gumption", "LaFruit", "Smith"}.
Write the method header and body for uniqueNames below.

```
public class Pool {

    PROBLEM 8 : (The biggest course)

    Complete the method emailsLargestCourse that has one ArrayList parameter named all. Each string
    in all has three pieces of information separated by colons: the course name, the name of a student and the
    student’s email. This method returns an ArrayList of the emails of all students registering for the course
    with the largest enrollment. Assume there is just one course with the largest enrollment.
    An example of the Strings in an ArrayList called dukeSpring2011 is below.

    CompSci 100:Fred Jack Smith:fjs@duke.edu
    History 117:Fred Jack Smith:fjs@duke.edu
    CompSci 102:Arielle Marie Johnson:amj@duke.edu
    CompSci 100:Arielle Marie Johnson:amj@duke.edu
    CompSci 006:Bertha White:bw@duke.edu
    Econ 051:Bertha White:bw@duke.edu
    English 112:Harry Potter:hp@duke.edu
    CompSci 100:Harry Potter:hp@duke.edu

    The call emailsLargestCourse(dukeSpring2011) would return the emails of "CompSci 100" (the largest
    course enrollment) in sorted order: ["amj@duke.edu", "fjs@duke.edu", "hp@duke.edu"]

    public ArrayList<String> emailsLargestCourse(ArrayList<String> all)
```

PROBLEM 9 : (Duke should be first)

For this question, you will write code to help sort basketball teams based on the total number of points they
have scored in a season. A team is represented by its String name. See the following class definition below:

```
public class BasketballPoints {
    // Maps from team name to total points scored
    private Map<String, Integer> teamPoints;

    /**
     * Initializes the teamPoints map
     */
    public BasketballPoints() {
        // TODO Part A
    }

    /**
     * Adds additional statistics to the map about a specific team.
     * Converts the shots array into the proper amount of points.
     * The team may or may not already be included in the map.
     *
     * @param name The team name
     * @param shots [free throws made (1-pointers),
```
* 2-pointers made, 3-pointers made
* */
public void addResult(String name, int[] shots) {
    // TODO Part B
}
/**
 * Uses the PointsComp class to sort the team names based on
 * total points, and returns the team name in the map which
 * has the highest point total.
 */
public String getBest() {
    // TODO Part D
}
/**
 * Comparator used to help sort the team names based
 * on their point totals stored in the map teamPoints.
 */
class PointsComp implements Comparator<String> {
    /**
     * Should rank teams by most points scored then team name
     */
    public int compare(String a, String b) {
        // TODO Part C
    }
}

A. The `teamPoints` map should map from team name to the total number of points scored by that team. A constructor should initialize any instance variables. Fill in the constructor for the `BasketballPoints` class below, creating a new instance of the map.

    /**
     * Initializes the teamPoints map
     */
    public BasketballPoints() {
    }

B. The `addResult` map should incorporate the results from a single game for a team and update the `teamPoints` map. The method is passed the team name and an array indicating the number of one-point shots made (free throws), the number of two-point shots made, and the number of three-point shots made.

    For example, `addResult("Duke",{19,22,9})` should add $1 \times 19 + 2 \times 22 + 3 \times 9 = 90$ points to Duke's overall point total stored in the map.

    Complete the `addResult` method below. Be sure to read the comments for any hints.

    /**
     * Adds additional statistics to the map about a specific team.
     * Converts the shots array into the proper amount of points.
     * The team may or may not already be included in the map.
     * */
C. You decide to rank teams by total score, so that teams with a higher total points scored appear before teams with a lower points score. If each team has scored the same number of points, you should order the teams lexicographically (i.e., alphabetically).

Use the `teamPoints` map to complete the `compare` method of the `PointsComp` comparator. Read the comments to make sure you return the proper values.

```java
/**
 * Comparator used to help sort the team names based
 * on their point totals stored in the map teamPoints.
 */
class PointsComp implements Comparator<String> {
    /**
     * a and b represent team names.
     * This method should return a negative number if a’s point total
     * is greater than b’s, a positive number if a’s
     * point total is less than b’s. If their point
     * totals are equal, it should return a negative number
     * if a is before b alphabetically, a positive number if b is
     * before a, and 0 if both the points and team names are equal.
     */
    public int compare(String a, String b) {
        // Implement the compare method according to the comments.
    }
}
```

D. Finally, complete the `getBest` method below, using the `PointsComp` comparator. Note that code has already been provided for you to convert the team names (the keys of `teamPoints`) into an `ArrayList`.

```java
/**
 * Uses the PointsComp class to sort the team names based on
 * total points, and returns the team name in the map which
 * has the highest point total.
 *
 * @return a String representing the team’s name
 */
public String getBest() {
    // Convert the keys of the teamPoints map to an ArrayList.
    List<String> names = new ArrayList<String>(teamPoints.keySet());
    // Implement the getBest method according to the comments.
}
```
PROBLEM 10: (TriPyramid (20 points))

The picture below shows a four-rowed, two-dimensional pyramid constructed of triangles – this will be called a 4-pyramid in this problem because it has four rows. The top triangle is number one, then the triangles are numbered left-to-right in a row and top-to-bottom as shown. In the fourth row there are seven triangles; in general in the \(N^{th}\) row there are \(2N - 1\) triangles. In answering questions below assume we’re discussing an \(N\)-pyramid with \(N\) rows for a large value of \(N\).

A. What is the exact value or number of the right-most triangle in the seventh row?

B. What is the exact value of the right-most triangle in the 40\(^{th}\) row?

C. What is the exact value of the left-most triangle in the 61\(^{st}\) row?

D. Using big-Oh what is the number of triangles in the bottom row of a pyramid with \(N\) rows. Justify your answer.

E. Using big-Oh what is the number of triangles in the bottom row of a pyramid with \(N^2\) total triangles. Justify your answer.

F. Using big-Oh what is the number of triangles in the bottom row of a pyramid with \(N^4\) total triangles. Justify your answer.

G. The diagram below shows four 3-pyramids combined to make a 6-pyramid. Assume there is a function or method \texttt{combine} that takes an \(N\)-pyramid as a parameter and returns a new pyramid created by combining four \(N\)-pyramids as shown. For example, the pyramid shown below would be returned by the call \texttt{combine(3)}.

I. What is the big-Oh number of triangles in the bottom row of the pyramid returned by the call \texttt{combine(N)}. Justify your answer.
II. What is the big-Oh value of the rightmost pyramid in the bottom row (the triangle with the largest number) in the pyramid returned by the call `combine(combine(N))`. Justify your answer.

III. Consider the pseudo-code below for a sequence of calls to create a pyramid. For example, when \( N = 2 \) the initial pyramid \( p \) is constructed before the loop with two rows and three triangles in the bottom row; the loop then executes twice. The first time through the loop results in \( p \) having four rows. The second time through the loop results in \( p \) having eight rows (with 15 triangles in the bottom row and the value printed is 64).

\[
\text{Pyramid } p = \text{new Pyramid}(N); \quad // \text{create pyramid with } N \text{ rows}
\]

\[
\text{int size = p.rows();} \quad // \text{set size to } N
\]

\[
\text{for(int } k=0; k < \text{size}; k++){}
\]

\[
\quad p = \text{combine}(p);
\]

\[
\}
\]

\[
\text{System.out.println("biggest number is "+p.lastPyramid());}
\]

If the value of \( N = 10 \) so that the initial pyramid has 10 rows in which the last pyramid is numbered 100 what is the value printed by the code above? Justify your answer. Your answer should be exact, but can be expressed using exponentiation and multiplication, e.g., \( 3^5 \times 100^2 \) is acceptable as an answer.

PROBLEM 11: (Reasoning About Lists)

The method `duplicate` in the code shown below changes parameter `list` so that it’s doubled in place – for example, the list

\[
(\text{"ape"}, \text{"bat"}, \text{"cat"}, \text{"dog"})
\]

is changed to the list below as a result of the call `duplicate(list)`.

\[
(\text{"ape"}, \text{"ape"}, \text{"bat"}, \text{"bat"}, \text{"cat"}, \text{"cat"}, \text{"dog"}, \text{"dog"})
\]

In this problem, the method `duplicate` is called with both an `ArrayList` and a `LinkedList` as parameters – the times for duplicating the different lists are shown tabularly and graphically below the code where the size of the list varies from 10,000 to 150,000 strings. The graphic is only for the `ArrayList` call to `duplicate` since the `LinkedList` timings are too small in comparison.

```java
public List<String> duplicate(List<String> list){
    ListIterator<String> iter = list.listIterator();
    while (iter.hasNext()){}
    String s = iter.next();
    iter.add(s);
}
    return list;
}
```

// code to call duplicate
for(int sz = 10000; sz <= 150000; sz += 10000){
    ArrayList<String> alist = createArrayList(sz);
    LinkedList<String> llist = createLinkedList(sz);
    // timing code not shown
    alist = duplicate(alist);
    llist = duplicate(llist);
    // print timings
}
A. Using the same code on the same computer how much time will it take to duplicate both an ArrayList and a LinkedList list with 1,000,000 (one million) values. Justify your answers (your answer will be considered approximate, we’re looking for close enough.) Don’t rely on the coefficients of the quadratic shown, use reasoning based on the empirical timings.

B. Using big-Oh, what is the complexity of duplicating both an N-element ArrayList and LinkedList using the code above. Justify your two answers empirically, by making explicit references to the timings.

C. We haven’t discussed ListIterator in class, but the code runs quickly for LinkedList because new elements are added to the list, in place, as the iteration over the list takes place. Based on the code and your understanding what is the runtime of the single statement iter.add(s) in the code above is for both LinkedList and ArrayList lists, justify your answer.

D. The code below is an alternate version of the duplicate code above. This code correctly modifies list so that it is doubled-in-place — its behavior is exactly the same as the code that uses the ListIterator, but the timings are different.

```java
public void duplicate2(List<String> list){
    int originalSize = list.size();
    list.addAll(list);
    for(int k=originalSize-1; k >= 0; k -= 1){
        String current = list.get(k);
        list.set(2*k, current);
        list.set(2*k+1, current);
    }
}
```

Explain why the for loop runs down to zero rather than up from zero (which would not work).

E. Provide a justification for why this code runs very quickly for ArrayList parameters and very slowly for LinkedList parameters. On the same computer on which the first timings were made, the LinkedList duplication takes 2.28 seconds for 20,000 strings and 18.7 for 40,000 strings whereas the ArrayList timings are all below 0.01 seconds.

Explain the timings by providing a reason for them based on the code and your understanding of how the classes LinkedList and ArrayList are implemented.