Data Structures

Name: _____ ID: _____

These questions do not have a formal, definitive answer. They are meant to be food for thoughts. Feel free to seek answers on browsing the Internet, talking to other software developers or reading books.

1. Our famous company, *Toy Inc.* would like to please its employees, by giving them prizes. In each case, tell me which data-structures you would use:

   (a) Each week the company will randomly choose an employee, who will receive a toy as a prize.

   (b) To ensure that each employee receives a toy, every time an employee is chose, his or her name is removed from the list of possible choices.

   (c) The company would like to name each toy after the name of one of the employees. Toys should not have repeated names.

   (d) The company would like to use the names in the order of frequency. That is, more frequent names should be used first.

   (e) The company would like to distribute tickets to soccer games among the employees. How to organize a waiting list?

2. In computer science we hear a lot about "data-structures". After all, what is this?

3. How many data-structures do you know? Could you list all of them? What are these data-structures good for?

4. Many languages provide libraries where data-structures are nicely implemented. Can you give examples?

5. The Java Collections Library is based on seven interfaces: `Collection`, `Set`, `SortedSet`, `List`, `Queue`, `Map` and `SortedMap`. What are the roles of each of these interfaces?

6. Which methods we would normally find in a collection? Why is Java `Collection` an interface, and not an implementation?

7. There are two ways, in Java, to access the elements stored in a collection. We have iterators, and the *for each* construct. Illustrate each of these concepts with two code examples.

8. When should we use iterators instead of the for each loop?

9. Sets contain three implementations: `HashSet`, `TreeSet`, and `LinkedHashSet`. What is the difference between them?

10. How to remove duplicates from a collection? And how to remove duplicates preserving the order of the elements in the original collection? Illustrate these ideas by coding a program that removes duplicates from the argument line.

11. Modify the program above to print a list of the duplicate words, and a list of the unique words.

12. Collections are implemented using generics. Illustrate the advantages of this approach by coding a method `swap` that swaps two elements of a generic list. Then, use the `swap` method to shuffle the elements of a list.

13. Lists, in Java Collections, have a special iterator, `ListIterator`. Show how to use this iterator by coding a program that goes over a list backwards.

14. Continuing with our examples of `ListIterator`, crank up a method that replaces all the occurrences of a value in a list for another value. Your method should have the signature:

```
public static <E> void replace(List<E> list, E val, E newVal)
```

15. How to code a method that deals a deck of cards? That is, this program receives a list of cards plus a hand size n, removes n elements from the deck, and returns these n elements. Use the following signature:

```
public static <E> List<E> dealHand(List<E> deck, int n)
```

16. What does the following program do? What is its time complexity?

```
static <E> List<E> heapSort(Collection<E> c) {
  Queue<E> queue = new PriorityQueue<E>(c);
  List<E> result = new ArrayList<E>();
  while (!queue.isEmpty())
    result.add(queue.remove());
  return result;
}
```

17. Implement a program that counts the frequency of each word in the argument line, and then prints the frequency in alphabetical order.