1.

Add(), Remove(), size(),clear(),~~compareTo(),~~ equals

**2. Which of the following classes use the Collection<t> interface**

ArrayList,  
TreeSet,  
HashSet  
LinkedList

They all implement the Collection interfaces: <https://docs.oracle.com/javase/7/docs/api/java/util/Collection.html>

3a **What are the primary difference between List<t> and Collection<t>**

List<T> implements Collection<t>. List<T> included additional methods that are related to getting\setting\removing indexes elements which is not part of the Collections interfaces.

**3b What is the primary difference between the Set<t> and Collection<T>**

Sets are designed to be collection of related items. Collections can be collection of unrelated items. Sets cannot have duplicate items, while collections can have more than one of same item.

**4. With respect to Comparable<T> and Comparor<T> interfaces:**

**a. What is the primary benefit of using the interfaces**

The primary benefit to using interfaces is to have a standard way to access the object that is common to all, but the actual implementation of the interface can different Customized code can be added that specically clarifies what is required for two object to be equal, rather than two objects being identical. Example a book object maybe be a hardback or a paper back. When determining two books are equal, the code may use that field to determine if two objects are equal.

**b) Name the methods that rea include in each of the interaces**

Comparable<T> : compareTo(object obj)   
Comparor<T>: int compare([T](https://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html) o1, [T](https://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html) o2)

Comparable<T> compares the current object to another object

Comparor<T> compares two objects, but does not compare against the current oject

**Algorithm Analysis**

**1 What is the input and output for a runtime function T(n)**

T is the output  
N is the input

2

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 3n+n | Logn ~~+n~~ |  |
| 2^n | ~~6~~nlogon |  |
| 7n^2 | ~~3~~n+n |  |
| 12^n + 13^n | ~~7~~n^2 | Growth rate n^2 |
| Logn +n | 12^n + 13^n =(15n^2) = n^2 | Growth rate n^2 |
| 3^n + 2^n | 2^n | Growth rate ^n |
| 6nlogon | 3^n + 2^n = 5^n | Growth rate ^n |

3 True false. It is possible that all of the following big-oh estimates could be true for a single algorithm with the runtime of T(0)….  
 Answer: True

4. Five the pseudo code for an algorithm called “join”

function join (list\_a, list\_b)  
 totalLength <- list\_a.length + list\_b.length  
 combinedArray <-new array[totalLenght]

index <- 0  
 for each element in list\_a  
 combinedArray[index] <- element  
 index <- index + 1  
   
 for each element in list\_b  
 combinedArray[index] <- element  
 index <- index + 1

return combinedArray

5 Give a big-oh analysis of your answer to 4  
   
 O(n). The function has to loop through each element in both arrays only once.

6. What do the following noations signifiy

NA…. ☹

List, Stacks, Queues, and interators

1. Give a tight big-oh estimate of the following opperations in an array list

|  |  |
| --- | --- |
| Add(element) | O(1) |
| Add(index, element) | O(n) |
| AddFirst(element) | O(n) |
| AddLast(element) | O(1) |
| Remove(index) | O(n) |
| RemoveFirst(index) | O(n) |
| RemoveLast(element) | O(n) |
| Get(index)set(index) | O (1) |

1. **What is an amortized running time?**

The average run time of the function between the worst case and the best case.

**Why us this important concept when discussing add method in an arraylist**

When adding at a specific index, the remaining items to the right need to be shifted. If its added as the first element,then the cost is higher than when compared to adding as the last element. The cost is determing the average runtime between the best and the worst.

1. **Why is this important when discussing add() method.**

When adding an element at an index, it requires a shift of all other elements in the internal array. With large array, the shift be costly, and compounded each time a new item is added.

1. **What are the benefits of iterator design pattern.**

Have an **iterator** in a class allows for the looping through elements using the foreach loop without requiring the comsumer of the class understanding the internal structure design of the class.

**Name the two interfaces that are used with iterator pattern**

First()

Next()

Gets the first item in collection, and then the next item in the collection.

**Why do iterators throw “*ConcurrentModificationExpection”* Objects**

It is thrown when trying to modify an existing element while iterating, such as attempting to add or remove an element.

**Search**

1. **What is the big-oh of the following search routines of the stated input types?**

|  |  |  |  |
| --- | --- | --- | --- |
| Seach Routine | Best | Worst | Average |
| Linear search | O(1) | O(n) | O(n/2) |
| Binary search | O(1) | O(n) | O(Log(n)) |
| Interp | O(1) | O(n) | O(n) |

1. State the halving principle.

Each time through the search loop half the elements cut in half until down to a single element which is either the element found, or not found.

3 give the following array and search for value 89

0,13,45,51,77,84,88,87,99  
 84,88,87,99  
 88,87,99  
 87  
  
 value not found

1. Explain how an approaite input array, the interpolation search can be an improvement over a binary search

NA… do not recall enough details about the interpolation search, but do recall it was discussed in class.

Hashing

1. Matching

hashcode= b

hashtable = c

collollision =d

load factor = e

linear/quad probing = f

chainging = a

2 what are the advantages\disadvantages of a high load factor.

Advantages: Uses less memory

Disadvantages: high Collison rate

Low load factor?

Basically just the reverse of high load factor. Less collisions, but uses more memory.

3 hash table

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| c | e |  | **b** | A | C | D | F |  |  |