The ArrayList and LinkedList classes each store a collection of data. Give one application where an ArrayList is the more appropriate container to use and one application where a LinkedList is the more appropriate container to use. Be sure to explain your reasoning.

Below is one possible implementation of the Lookup.add(E o) method.

```
public boolean add(E o) {
  boolean exists = false;
  boolean retVal = false;

  for (int i=0; i < size() && !exists; i++) {
    E member = get(i);

    if (o.equals(member)) {
      exists = true;
    }
  }
  if (!exists) {
    retVal = true;
    super.add(o);
  }

  return retVal;
}
```

(a) (8 points) Assume that all calls in add take a fixed amount of time regardless of the size of the set (e.g., size() takes x ms regardless of the value returned by the method). Give the asymptotic time complexity for the add method. Be sure to justify your answer.

(b) (2 points) Assume that all calls in add, except size(), take a fixed amount of time regardless of the size of the set and that size() is $O(n)$ where n is the number of elements in the set. Give the asymptotic time complexity for the add method. Be sure to justify your answer.

Below is a slightly incorrect implementation of the SinglyLinkedList.toArray(T[] array) method.

```
public <T> T[] toArray(T[] array) {                                    // c1
  array = (T[]) new Object[size()];                                    // c2
  int index=0;                                                         // c3
  for(Node<E> current=front; current!=null; current = current.next)    // c4
  {                                                                    // c5
    array[index] = (T)current.element;                                 // c6
    ++index;                                                           // c7
  }                                                                    // c8
  return array;                                                        // c9
}                                                                      // c10
```

(a) (7 points) Indicate the number of times each line will be executed (as a function of n — the number of elements in the collection).

(b) (3 points) Assume that all calls in toArray, except size(), take a fixed amount of time regardless of the size of the set and that size() is $O(n)$ where n is the number of elements in the set. Give the asymptotic time complexity for the add method. Be sure to justify your answer.

Use the ArrayPureStack<E> and/or CircularPureQueue<E> classes to implement a `static` method called reverse that accepts an array of floating point numbers and returns an array of floating point numbers in reverse order of the array passed in.

Suppose a binary tree exists and that the value of each node obeys the following ordering: $left.value < value < right.value$ (note: no two nodes can have the same value). Implement the following method:

```
// If a node with the value target is found, that node is returned
// otherwise, the method returns null
protected Node<E> find(Node<E> disaster, E target)
{
```

}

Recall each node had four fields: Node<E> parent, Node<E> left, Node<E> right, and E value.

What is the Map.Entity? How is it used in the Map interface?

Here is the dotProduct method we discussed in class yesterday.

```
public double dotProduct(SVector<V> vec)
{
  Set<Map.Entry<Integer, V>> setView;
  Map<Integer, V> map;
  if(size()<vec.size())
  {
    setView = data.entrySet();
    map = vec.data;
  }
  else
  {
    setView = vec.data.entrySet();
    map = data;
  }
  double dProduct = 0;
  for(Map.Entry<Integer, V> entry : setView)
  {
    if(map.containsKey(entry.getKey()))
    {
      dProduct += (Double)map.get(entry.getKey())*(Double)entry.getValue();
    }
  }
  return dProduct;
}
```

a) What is the purpose of the first `if` statement?

b) The data field used in lecture was a SortedMap<Integer, V> data = new TreeMap<Integer, V>();. Suppose the field was changed to: Map<Integer, V> data = new HashMap<Integer, V>();. What, if any, problems would arise with the dotProduct method?