HW1 Solutions

1) a) 3/2
b) $2n^2 + 3n$
c) 6n

2) Abstraction is the separation of a high-level idea about some thing (e.g. a data structure) and the low-level details concerning precisely how that thing works.
An abstract data type (ADT) like the SackADT interface, describes the high-level functionality of the type, such as adding and removing items from a collection, or getting the state of the collection (empty vs. non-empty). On the other hand, the implementation of the sack would contain code that handles adding and removing elements, including a potentially large amount of code for handling things like edge cases (e.g. adding to a list that is currently full) and various possible errors in input (e.g. adding a null value to the collection).

3) Advantage of using an array is the ability to access any arbitrary element in constant time.
Downside of using an array is that its capacity is fixed at the time it is created.

4)

```
public void removeDuplicates(ArrayGSackIterable<Integer> sack) {
      for (Iterator<Integer> itr = sack.iterator(); itr.hasNext();) {
            int currval = itr.next();
            boolean foundOne = false;
            for (Iterator<Integer> itr2 = sack.iterator(); itr2.hasNext();) {
                  int compval = itr2.next();
                  if (currval == compval) {
                        if (foundOne == false) foundOne = true;
                              else itr2.remove();
                  }
            }
      }
}
```

5) The code appends the contents of src to the end of dst.

6)

a)

after String s = myList.get(0), myList = [5, 4, 3, 2, 1]
after myList.add(0, myList.get(2)), myList = [3, 5, 4, 3, 2, 1]
after myList.add(2, s), myList = [3, 5, 5, 4, 3, 2, 1]

b)

```
void swap(int I, int j, List<String> myList) {
      String s = myList.get(i);
      String e = myList.get(j);
      myList.remove(i);
      myList.add(i, e);
      myList.remove(j);
      myList.add(j, s);
}
```

EC)

```
public class CartesianProduct{
          public static List<List<String>> cartProdStrings(List<String> A, List <String>
B) {
                List<List<String>> cartProd = new ArrayList < List <String>>() ;
                for ( String aStr : A ) {
                      for ( String bStr : B ) {
                      // Add this pair to the Cartesian product set
                            cartProd.add ( Arrays.asList (aStr,bStr)) ;
                      }
                }
                return cartProd;
                }
                public static void main ( String [] args ) {
                      List <String> A = Arrays.asList("A0" , "A1" , "A2") ;
                      List < String > B = Arrays . asList ( " B0 " , " B1 " ) ;
                      List < List < String > > cartProd = cartProdStrings(A,B) ;
                      System.out.println (" The Cartesian product set : ") ;
                      System.out.println (cartProd.toString()) ;
                }
          }
```