Throughout this test, assume that the following classes and methods are available. These classes are taken
directly from the material used in class. There should be no methods you have never seen before here.
However, if you know of additional methods not listed here, you may use any of them in your solutions.

**Integer**

```java
class Integer implements Comparable {
    // The smallest and largest values of type int
    public static final int MIN_VALUE
    public static final int MAX_VALUE

    // Returns the integer represented by the
    // argument as a decimal integer.
    public static int parseInt(String s)

    // Returns a new String object representing
    // the specified integer.
    public static String toString(int i)

    // Returns value of Integer object as an int
    public int intValue()

    // Returns true if obj is an Integer object
    // that contains same int value as this object.
    public boolean equals(Object obj)

    // Returns 0 if other is equal to this Integer;
    // a value less than 0 if this Integer is less
    // than other; and a value greater than 0 if
    // this Integer is greater than other
    public int compareTo(Integer other)
}
```

**Object**

```java
class Object {
    // Returns true if o is the same as this object
    boolean equals(Object o)

    // Returns string representation of this object
    String toString()
}
```

**Iterator**

```java
class Iterator {
    // Returns true if iteration has more elements.
    boolean hasNext()

    // Returns next element in this iteration.
    Object next()

    // Removes current element from the collection.
    void remove()
}
```

**array**

```java
class array {
    // number of objects stored in the array
    public int length;
}
```

**Comparable**

```java
interface Comparable {
    // Returns a positive value if this object is
    // greater than the given one, 0 if this object
    // is equal to the given one, and a negative
    // value if this object is less than the given one.
    int compareTo(Object other)
}
```

**Scanner**

```java
class Scanner implements Iterator {
    // Create Scanner that reads data from a file.
    public Scanner(File file)

    // Create Scanner that reads data from a string.
    public Scanner(String str)

    // Change delimiters used to separate items
    public void useDelimiter(String characters)

    // Check if more items are available
    public boolean hasNext()

    // Get next delimited item as a string
    public String next()

    // Get next delimited item as an integer value
    public int nextInt()

    // Get next delimited item as a real value
    public int nextLong()
}
```

**String**

```java
class String {
    // Returns the length of this string.
    public int length()

    // Returns a new string that is a substring
    // of this string. The substring begins at
    // the specified beginIndex and extends to
    // the character at index endIndex - 1.
    public String substring(int beginIndex, int endIndex)

    // Returns the index within this string of
    // the first occurrence of str
    public int index0f(String str)

    // Returns the index within this string of the
    // first occurrence of str after index start
    public int index0f(String str, int start)

    // Returns the index within this string of
    // the last occurrence of str
    public int lastIndexOf(String str)
}
```
**Map**

```java
public class Map<KEY, VALUE>
{
    // Returns the number of keys in this map.
    public int size ()
    {
    }

    // Associates given value of type VALUE with
    // given key of type KEY in map.
    public VALUE put (KEY key, VALUE value)
    {
    }

    // Returns value of type VALUE associated with
    // given key of type KEY.
    public VALUE get (KEY key)
    {
    }

    // Removes mapping for given key of type KEY
    // from map, returning old value of type VALUE
    public VALUE remove (KEY key)
    {
    }

    // Returns set of keys in map
    public Set<KEY> keySet ()
    {
    }

    // Returns collection of values in map
    public Collection<VALUE> values ()
    {
    }
}
```

**Set**

```java
public class Set<ITEM>
{
    // Returns the number of elements in this set.
    public int size ()
    {
    }

    // Returns true if this set contains o of type ITEM
    public boolean contains (ITEM o)
    {
    }

    // Adds the given element of type ITEM to this set
    // if it is not already present.
    public boolean add (ITEM o)
    {
    }

    // Adds the given collection of elements of type
    // ITEM to this set if they are not already present.
    public boolean addAll (Collection<ITEM> c)
    {
    }

    // Retains only the elements in this set that
    // are also contained in the given collection
    public boolean retainAll (Collection<ITEM> c)
    {
    }

    // Removes given element or type ITEM from this set
    public void remove (ITEM o)
    {
    }

    // Returns an iterator over the elements in this set.
    // The elements are returned in ascending order.
    public Iterator<ITEM> iterator ()
    {
    }
}
```

**List**

```java
public class List<ITEM>
{
    // Returns the number of elements in this list.
    public int size ()
    {
    }

    // Searches for the first occurrence of the given
    // element of type ITEM, returns -1 if not found
    public int indexOf (ITEM item)
    {
    }

    // Returns element of type ITEM at index in list.
    public ITEM get (int index)
    {
    }

    // Removes element of type ITEM at index in list;
    // shifts any subsequent elements to the left (i.e.,
    // subtracts one from their indices).
    public ITEM remove (int index)
    {
    }

    // Removes first instance of element of type ITEM
    // in list, shifts any subsequent elements to the
    // left. Returns false if o was the only instance
    // of that object in the list.
    public boolean remove (ITEM o)
    {
    }

    // Appends given element of type ITEM to end of list.
    public boolean add (ITEM o)
    {
    }

    // Appends all elements from c to end of list.
    public boolean addAll (Collection<ITEM> c)
    {
    }

    // Replaces element at the given position in this
    // list with the given element of type ITEM.
    public boolean set (int index, ITEM c)
    {
    }

    // Returns an iterator over the elements in
    // this list. The elements are returned in
    // the order they were added.
    public Iterator<ITEM> iterator ()
    {
    }
```