Midterm Practice

1. Provide specification-based testing (partition and boundaries) for the following problem:

```java
    private static final String SPACE = " ";
    private static boolean isEmpty(final CharSequence cs) {
        return cs == null || cs.length() == 0;
    }
    /**
     * Left pad a String with a specified String.
     *
     * Pad to a size of {@code size}.
     *
     * @param str  the String to pad out, may be null
     * @param size  the size to pad to
     * @param padStr  the String to pad with, null or empty treated as single space
     * @return left padded String or original String if no padding is necessary,
     *  {@code null} if null String input
     */
    public static String leftPad(final String str, final int size, String padStr) {
        if (str == null) {
            return null;
        }
        if (isEmpty(padStr)) {
            padStr = SPACE;
        }
        final int padLen = padStr.length();
        final int strLen = str.length();
        final int pads = size - strLen;
        if (pads <= 0) {
            return str; // returns original String when possible
        }

        if (pads == padLen) {
            return padStr.concat(str);
        } else if (pads < padLen) {
            return padStr.substring(0, pads).concat(str);
        } else {
            final char[] padding = new char[pads];
            final char[] padChars = padStr.toCharArray();
            for (int i = 0; i < pads; i++) {
                padding[i] = padChars[i % padLen];
            }
            return new String(padding).concat(str);
        }
    }
```

```
Individual partitions (most interesting/ select):
- str parameter
Null
Empty string
Non-empty string (single or multiple characters)
- size parameter
Negative number
Positive number
- padStr parameter
Null
Empty
Non-empty (single or multiple characters)
- str,size parameters
size = len(str)
size < len(str)
size > len(str)

Combine partitions (most interesting/ select):
str is null
str is empty
size is negative
padStr is null
padStr is empty
size < len(str)
size > len(str)
the length of padStr is equal to the remaining spaces in str.
the length of padStr is greater than the remaining spaces in str.
the length of padStr is smaller than the remaining spaces in str

Boundaries (most interesting / select):
- size being precisely 0
- str having length 1
- padStr having length 1
- size being precisely the length of str
```

2. Provide specification-based testing (partition and boundaries) for the following problem:

```
/**
 * Finds the index of the given value in the array starting at the given index.
 *
 * This method returns INDEX_NOT_FOUND (-1) for a null
 * input array.
 *
 * A negative startIndex is treated as zero. A startIndex larger than the array
 * length will return INDEX_NOT_FOUND (-1).
 *
 * @param array
 *            the array to search through for the object, may be null
 * @param valueToFind
 *            the value to find
 * @param startIndex
 *            the index to start searching at
 * @return the index of the value within the array, INDEX_NOT_FOUND
 *         (-1) if not found or {@code null} array input
 */
public static int indexOf(final int[] array, final int valueToFind,
                                                int startIndex) {
    ...
}
```

```
Individual partitions (most interesting/ select):
- array
Null
Empty
Single
Multiple (repeats and no repeats)
- valueToFind
in the array
not in the array
- startIndex
negative
larger than array length
equal to the array length - 1 (last element).
before the index of the value to find
same as the index of the value to find
greater than the index of the value to find


Com bined partitions (most interesting / select):
- output -1:
array is null
empty array
value not in the array
startIndex larger than array length
startIndex after valueToFind
index exactly array.length-1  (value is in the array,but not the last)

- output a valid index (value is in the array for all)
negative start index
startValue before valueToFind
startIndex exactly valueToFind
index exactly array.length-1
multiple ocurrences
value is the first element and index is positive
value is the first element and index is negative

Boundaries (most interesting / select):
- index exactly array.length-1  (value is in the array,but not the last
        or it is the last)
- startIndex exactly valueToFind, before and after.
```