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);
        }
    }
```

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

```java
/**
 * 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) {
    ...
 }
```