SAT: Property-based Testing

1. Write property-based tests for the following procedure:

```java
public String sameEnds(String string) {
  int length = string.length();
  int half = length / 2;
  String left = "";
  String right = "";
  int size = 0;
  for (int i = 0; i < half; i++) {
    left = left + string.charAt(i);
    right = string.charAt(length - 1 - i) + right;
    if (left.equals(right)) {
      size = left.length();
    }
  }
  return string.substring(0, size);
}
```

```
- generator: end and middle (two strings)
- post-processing: end+middle+reverse(end) to get the string arguement
- validation: sameEnds result is end
```

2. Write property-based tests for the following procedure:

```java
public static boolean isLeapYear(int year) {
    if (year % 400 == 0)
        return true;
    if (year % 100 == 0)
        return false;
    return year % 4 == 0;
}
```

```
- generator: multiple of 400, multiple of 100, multiple of 4,
not a multiple of 4
- no post-processing
- validation: isLeapYear returns true,false,true,false in this order
for the 4 cases.
```

3. Write property-based tests for the following procedure:

```
public enum TYPE {
    NONE,
    SCALENE,
    ISOSCELES,
    EQUILATERAL
}

public static TYPE checkTriangle(int s1, int s2, int s3) {
    if (!isValidTriangle(s1, s2, s3))
        return TYPE.NONE;
    else if (isEquilateral(s1, s2, s3))
        return TYPE.EQUILATERAL;
    else if (isIsosceles(s1, s2, s3))
        return TYPE.ISOSCELES;
    else
        return TYPE.SCALENE;
}
public static boolean isValidTriangle(int a, int b, int c) {
    return (a + b > c && b + c > a && c + a > b);
}
public static boolean isEquilateral(int a, int b, int c) {
    return (a == b && b == c);
}
public static boolean isIsosceles(int a, int b, int c) {
    return (a == b || b == c || a == c);
}
```

```
- generator: one positive int (equilateral),
two positive ints that make a triangle (isosceles),
three positive ints that make a triangle (scalene),
three positive ints that violate either of the conditions for it to be
a triangle (none)
- no post-processing
- validation: checkTriangle returns the correct enumeration type
```