Stack Problems

A ***stack*** is an abstract data type (ADT) that is relatively easy to understand. Think about stacking your lunch trays in the cafeteria – the first person to place their tray on the stack will be at the bottom and will therefore be the last one "out" of the stack (cleaned by the cafeteria staff).

In this way, a stack of items is a "First in, Last out" (FILO) data structure. It is important to note that there is no stack data structure inside your computer (hence the ***abstract*** part of ADT) – a stack is a "wrapper", built around an already implemented structure (e.g. an array).

1. Create a **StackProbsRunner.java** class with a `main` method and a class called **StackProbs.java** that will define the methods below. Test the methods from Runner.

    a. Add the following helper method to the Runner class, which will construct a Stack from an array (for testing purposes, rather than multiple calls to `push`):

```
public static Stack<Integer> makeStack(int[] nums) {
        Stack<Integer> stack = new Stack<>();
        for (int num : nums)
                stack.push(num);
        return stack;
}
```

2. Complete the method `Stack<Integer> doubleUp(Stack<Integer> nums)` that replaces every value in the supplied stack with two of the same value (in the same order). For the first 2 methods you may use additional Stack objects for these problems, but no other data structure (e.g. List or array) or enhanced for loops. Try to avoid using other data structures for the last 2 methods.

```
(1, 3, 5, 0, -1) >>> (1, 1, 3, 3, 5, 5, 0, 0, -1, -1)
```

3. Complete the method `Stack<Integer> posAndNeg(Stack<Integer> nums)` that will split `nums`, such that all the negative numbers will be at the bottom of the stack, and all non-negative numbers will be at the top of the stack. The actual order of the elements is irrelevant if the negative and non-negative numbers are properly split.

```
(2, 9, -4, 3, -1, 0, -6) >>> (-4, -1, -6, 2, 9, 3, 0)
```

4. Complete the method `Stack<Integer> shiftByN(Stack<Integer> nums, int n)` that "shifts" `n` values from the bottom of the stack to the top of the stack. You can assume `n` will be a valid value (greater than 0, and not larger than the number of elements in the stack).

```
(7, 23, -7, 0, 22, -8, 4, 5), 3 >>> (0, 22, -8, 4, 5, 7, 23, -7)
```

5. Complete the method `String reverseVowels(String str)` that will reverse all the vowels (a, e, i, o, u) in a String. You must use a Stack in your solution. **Note**: there is an implicit cast when using compound operators (e.g. `+=`); you can concatenate a `char` or a Character to a String.

```
reverseVowels("hello how are you") >>> " hullo hew aro yoe"
```