Question: Leet code for next permutation.

Solution:

```
class Solution {
    public void nextPermutation(int[] nums) {
        final int n = nums.length;


    // From back to front, find the first number < nums[i + 1].
    int i;
    for (i = n - 2; i >= 0; --i)
      if (nums[i] < nums[i + 1])
        break;


    // From back to front, find the first number > nums[i], swap it with
    // nums[i].
    if (i >= 0)
      for (int j = n - 1; j > i; --j)
        if (nums[j] > nums[i]) {
          swap(nums, i, j);
          break;
        }


    // Reverse nums[i + 1..n - 1].
    reverse(nums, i + 1, n - 1);
  }


  private void reverse(int[] nums, int l, int r) {
    while (l < r)
```

```
      swap(nums, l++, r--);
  }


  private void swap(int[] nums, int i, int j) {
    final int temp = nums[i];
    nums[i] = nums[j];
    nums[j] = temp;
    }
}
```

Question: Leet code for Basic calculator.

Solution:

```
class Solution {
    public int calculate(String s) {
    int ans = 0;
    int num = 0;
    int sign = 1;
    // stack.peek() := the current environment's sign
    Deque<Integer> stack = new ArrayDeque<>();
    stack.push(sign);


    for (final char c : s.toCharArray())
      if (Character.isDigit(c))
        num = num * 10 + (c - '0');
      else if (c == '(')
        stack.push(sign);
      else if (c == ')')
        stack.pop();
      else if (c == '+' || c == '-') {
```

```
            ans += sign * num;
            sign = (c == '+' ? 1 : -1) * stack.peek();
            num = 0;
          }


        return ans + sign * num;


        }
    }
```