题目描述：

给一个正整数 NUM1，计算出新正整数 NUM2，NUM2 为 NUM1 中移除 N 位数字后的结果，需要使得 NUM2 的值最小。

输入描述：

1.输入的第一行为一个字符串，字符串由 0-9 字符组成，记录正整数 NUM1，NUM1 长度小于 32。

2.输入的第二行为需要移除的数字的个数，小于 NUM1 长度。

如：

2615371

4

输出描述：

输出一个数字字符串，记录最小值 NUM2。

如：131

示例 1

输入：

```
2615371
4
```

输出：

```
131
```

说明：

移除 2、6、5、7 这四个数字，剩下 1、3、1 按原有顺序排列组成 131，为最小值

```
import java.util.Scanner;
import java.util.Stack;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNext()) {// 注意，如果输入是多个测试用例，请通过 while 循环处理多
```

个测试用例

```
                String origin = in.nextLine();
                int b = in.nextInt();

                Stack<Integer> stack = new Stack<>();
                for (char c : origin.toCharArray()) {
                    int num = c - '0';
                    // 如果当前值比栈顶的小，则弹出栈顶数字
                    while (!stack.isEmpty() && stack.peek() > num && b > 0) {
                        stack.pop();
                        b --;
                    }
                    stack.push(num);
                }
                // 可能最大的数字在后面，导致没有移除足够数字，那么出栈直到 b=0
                while (b > 0) {
                    stack.pop();
                    b --;
                }

                StringBuilder strBuilder = new StringBuilder();
                while (!stack.isEmpty()) {
                    strBuilder.append(stack.pop().toString());
                }
                String result = strBuilder.reverse().toString();
                int index = 0;
                while (index < result.length() && result.charAt(index) == '0') {
                    index ++;
                }
                System.out.println(index != result.length() ? result.substring(index) : 0);
            }
        }
}
```