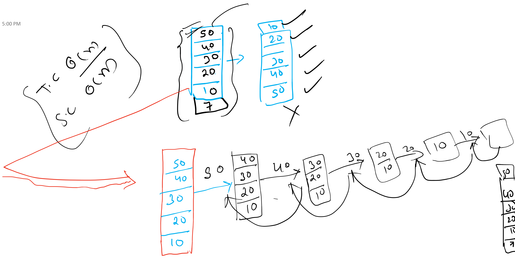
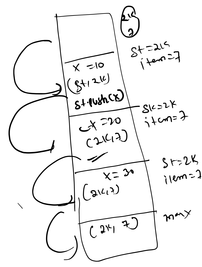
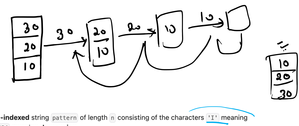
Lec-26

22/03/25 5:00 PM



```
public static void add_Last(Stack<Integer> st, int item) {
    if (st.isEmpty()) {
        st.push(item);
        return;
    }
    int x = st.pop();
    add_Last(st, item);
    st.push(x);
}
```



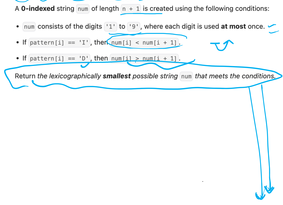


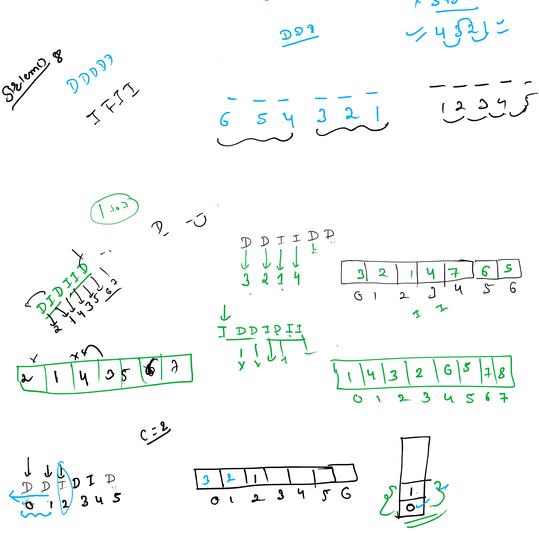
You are given a **0-indexed** string `pattern` of length `n` consisting of the characters `'I'` meaning **increasing** and `'D'` meaning **decreasing**.

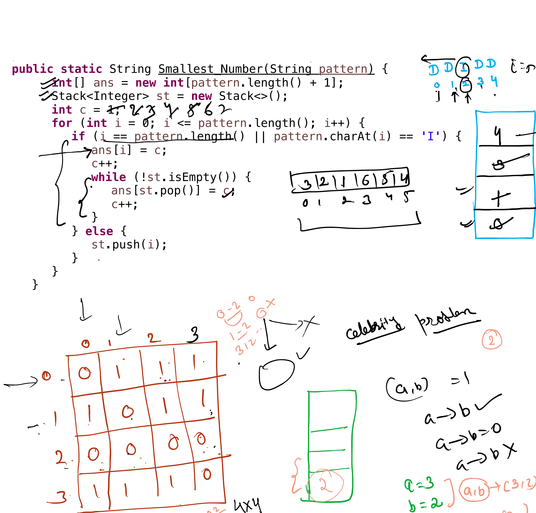
A **0-indexed** string `num` of length `n + 1` is created using the following conditions:

- `num` consists of the digits `'1'` to `'9'`, where each digit is used **at most** once.
- If `pattern[i] == 'I'`, then `num[i] < num[i + 1]`.
- If `pattern[i] == 'D'`, then `num[i] > num[i + 1]`.

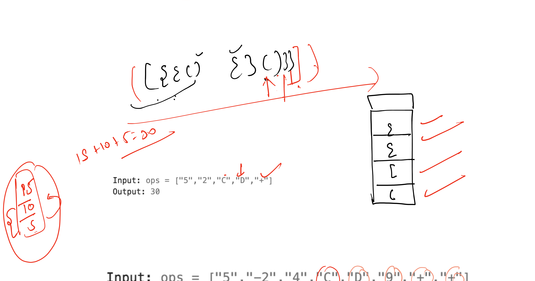
Return *the lexicographically* ***smallest*** *possible string* `num` *that meets the conditions.*







```
public static String Smallest_Number(String pattern) {
    int[] ans = new int[pattern.length() + 1];
    Stack<Integer> st = new Stack<>();
    int c = 1;
    for (int i = 0; i <= pattern.length(); i++) {
        if (i == pattern.length() || pattern.charAt(i) == 'I') {
            ans[i] = c;
            c++;
            while (!st.isEmpty()) {
                ans[st.pop()] = c;
                c++;
            }
        } else {
            st.push(i);
        }
    }
}
```

```
Input: ops = ["5","2","C","D","+"]
Output: 30
```

```
Input: ops = ["5","-2","4","C","D","9","+","+"]
Output: 27
```