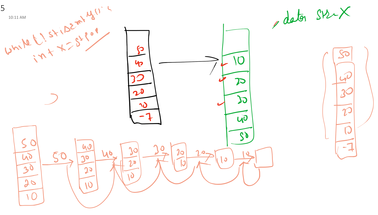
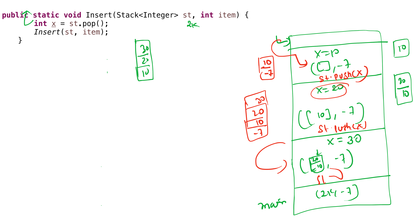
Lec-25
29/09/24 10:11 AM

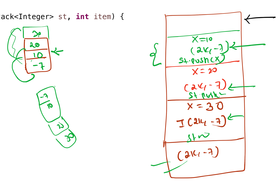




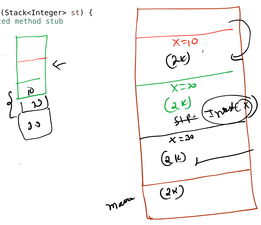
```
public static void Insert(Stack<Integer> st, int item) {
    int x = st.pop();
    Insert(st, item);
}
```



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



```
private static void Reverse(Stack<Integer> st) {
    // TODO Auto-generated method stub
    if (st.isEmpty()) {
        return;
    }
    int x = st.pop();
    Reverse(st);
    st.push(x);
}
```

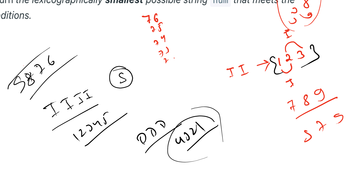


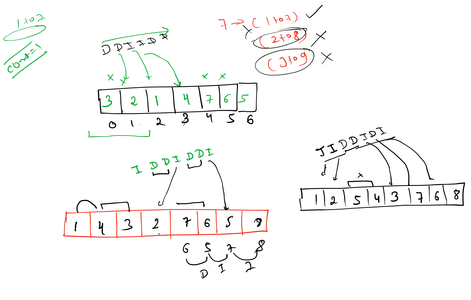
You are given a **0-indexed** string `pattern` of length `n` consisting of the characters `'I'` meaning **increasing** and `'D'` meaning **decreasing**.

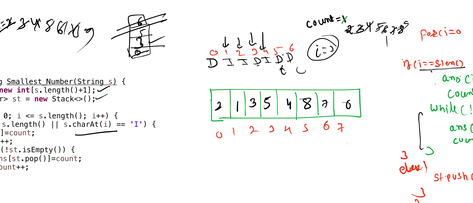
A **0-indexed** string `num` of length `n + 1` is created using the following conditions:

- `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]`.

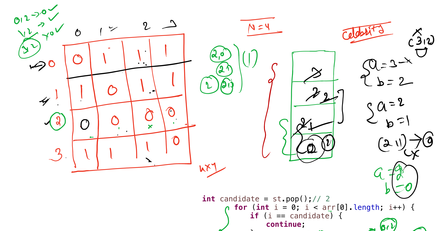
Return *the lexicographically **smallest** possible string* `num` *that meets the conditions.*







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



```
int candidate = s.pop();// 2
for (int i = 0; i < arr[0].length; i++) {
    if (i == candidate) {
        continue;
    }
    if (arr[candidate][i] == 1) {
        return -1;
    }
}
```

Example 2:

Input: s = "([])"

Output: true