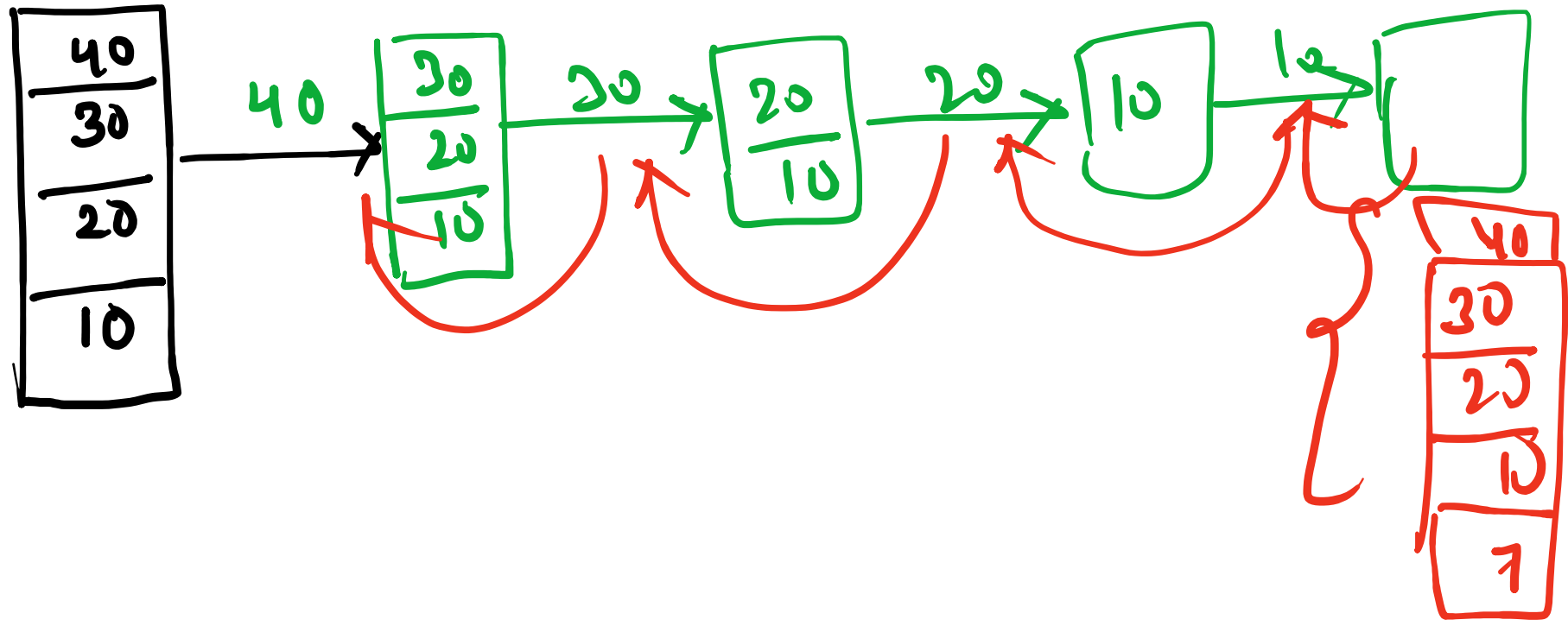
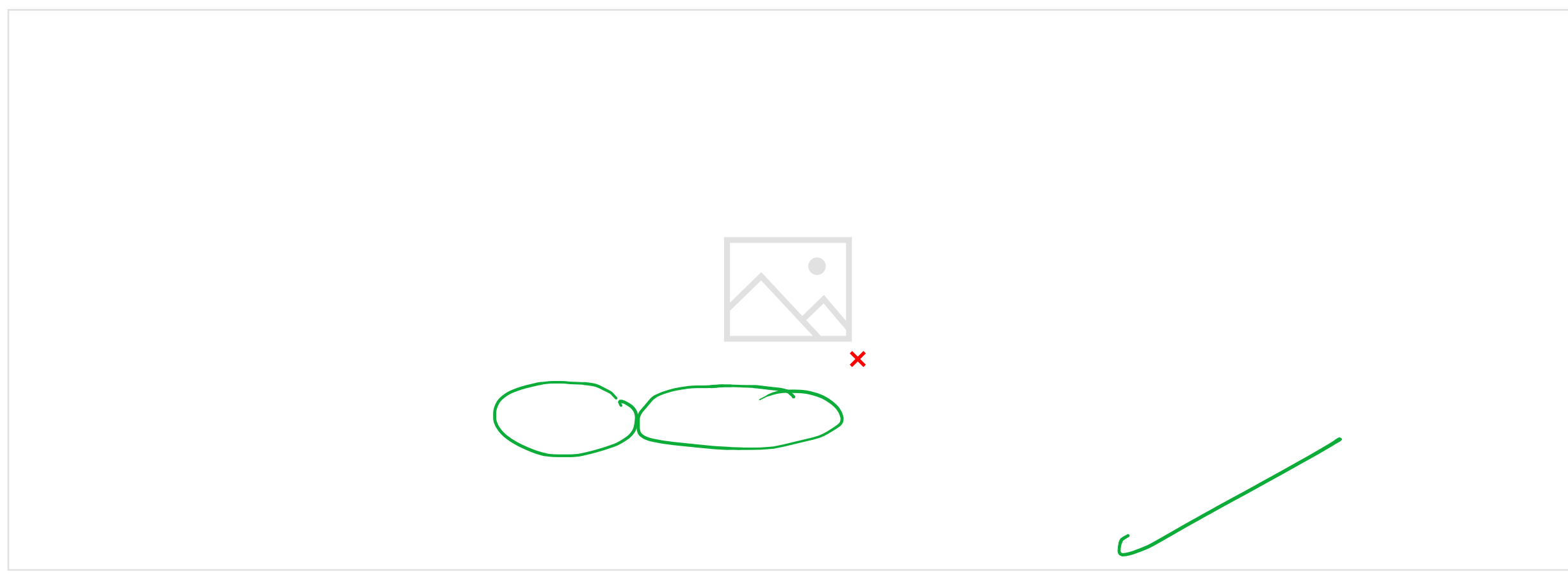
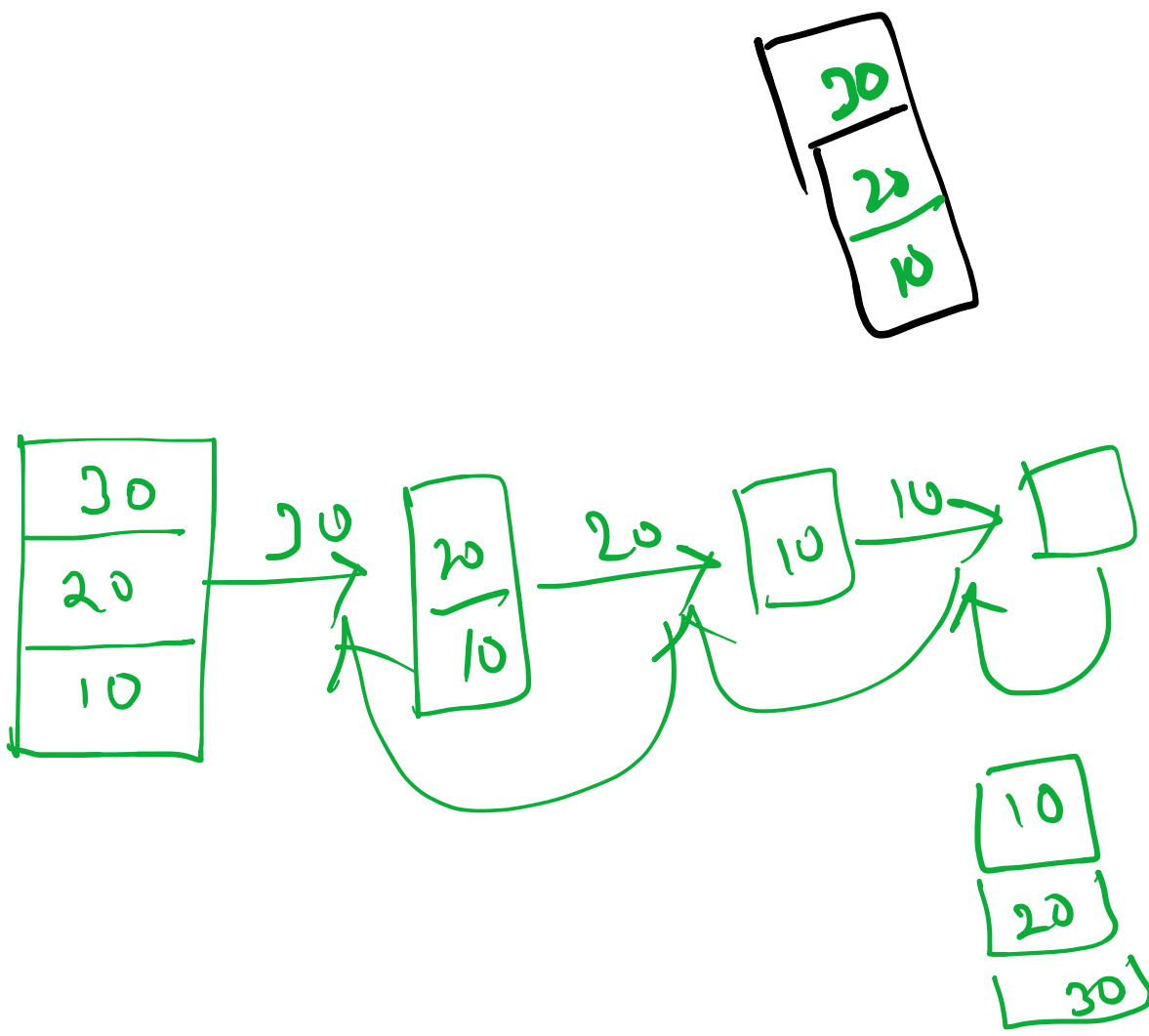
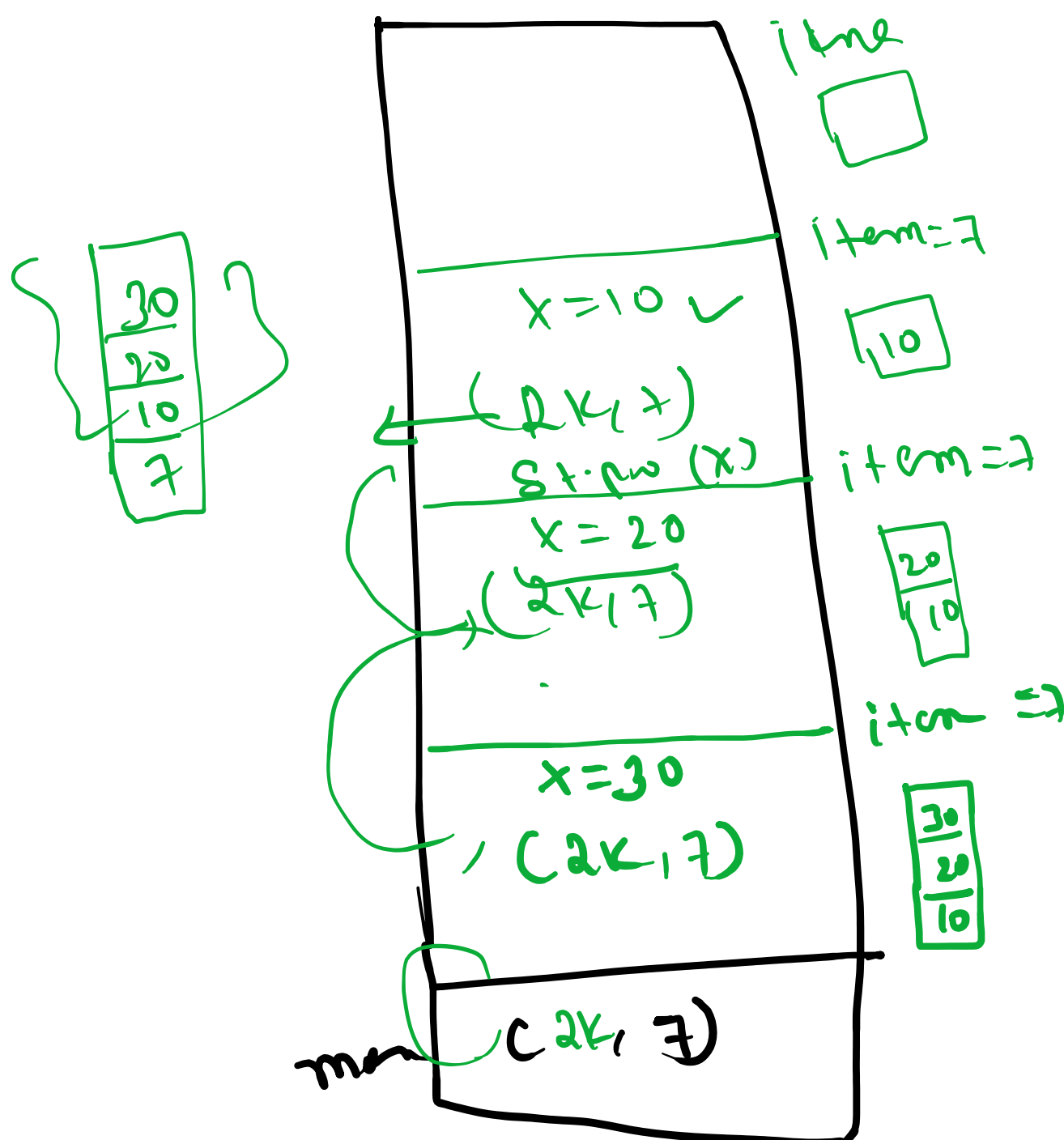


item=7



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



DP/
8 7 4
8 4 2
9 7 2
9 6 5
3 2 1

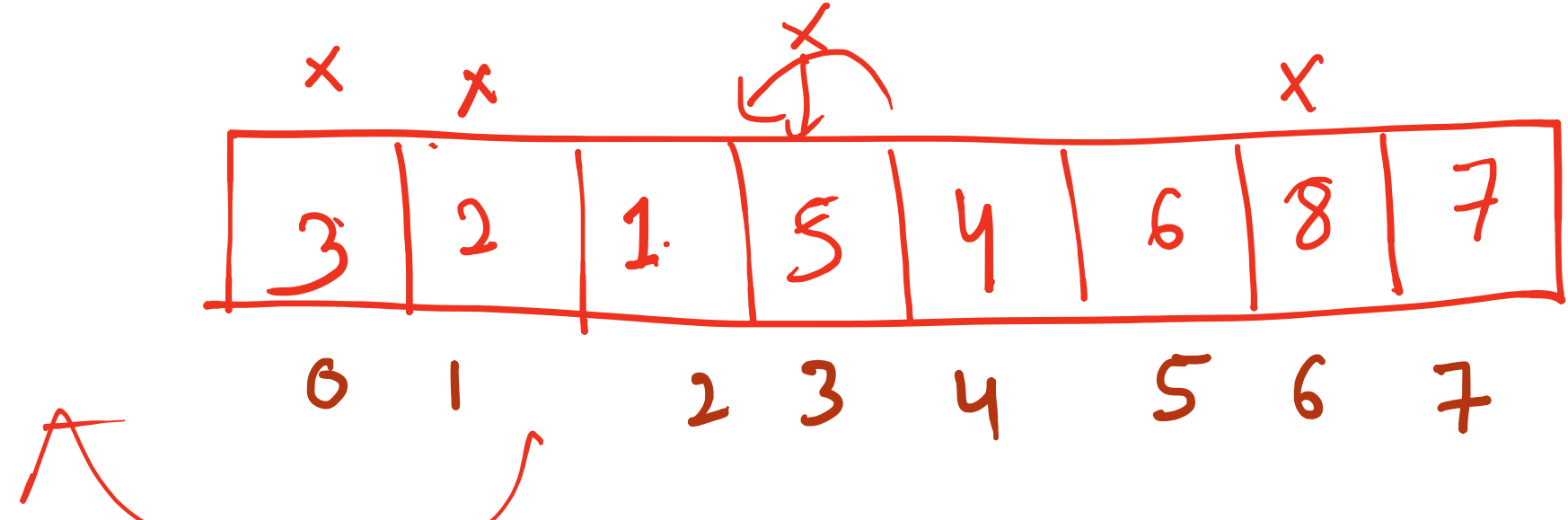
I I I
1 2 3 4
1 2 3 4

I F F I I
1 2 3 4 5 6

D D D
4 3 2 1

D D I D I D
1 2 3 4 5 6 7 8

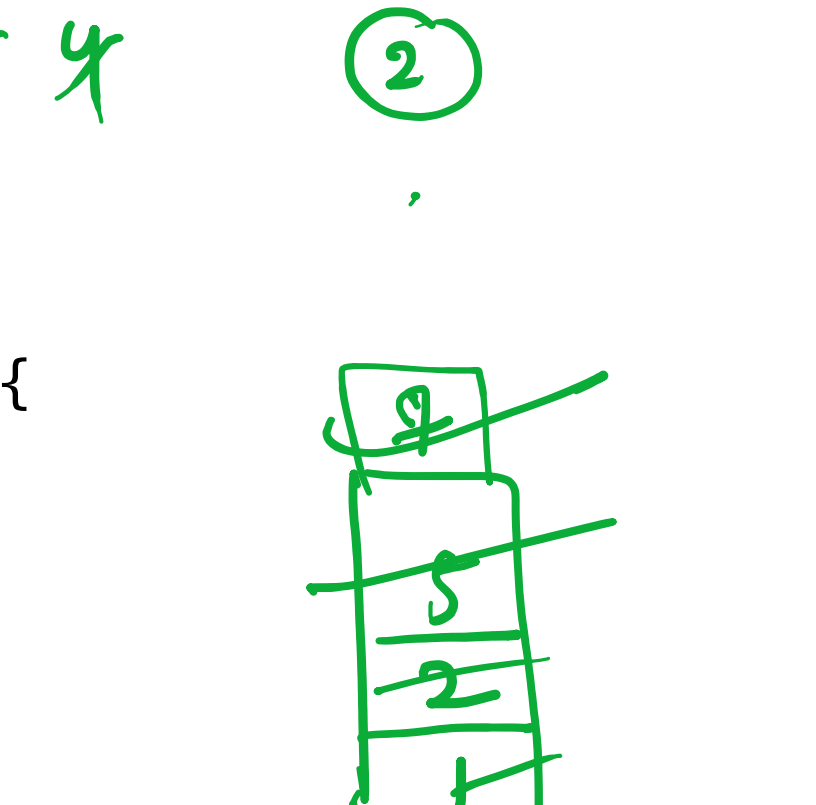
I D D I I
1 4 3 2 5 6



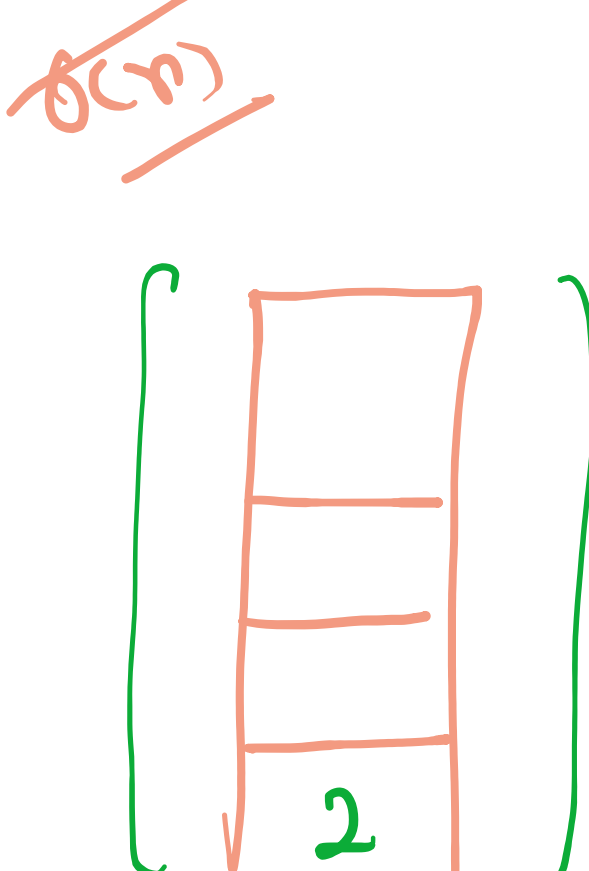
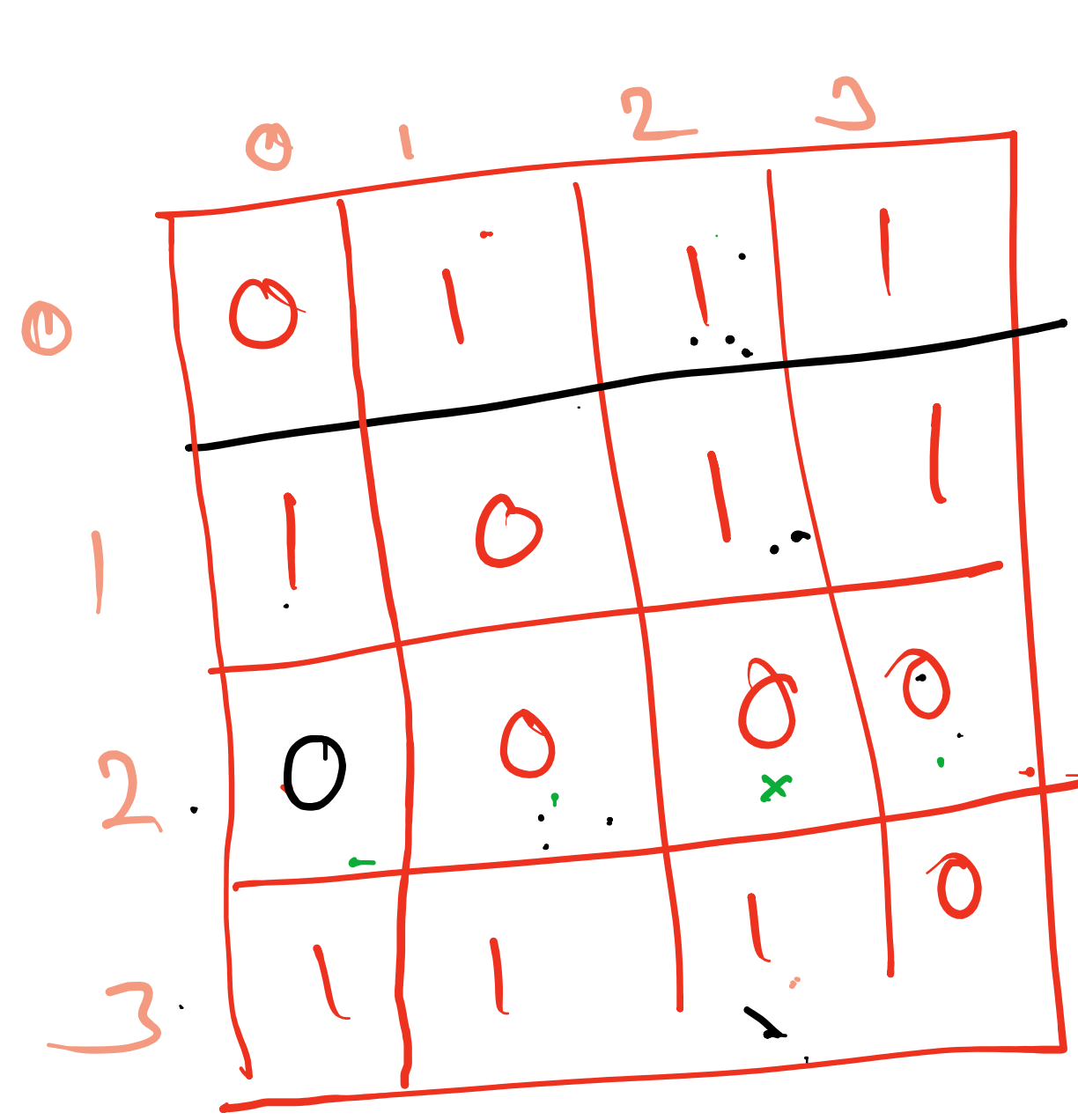
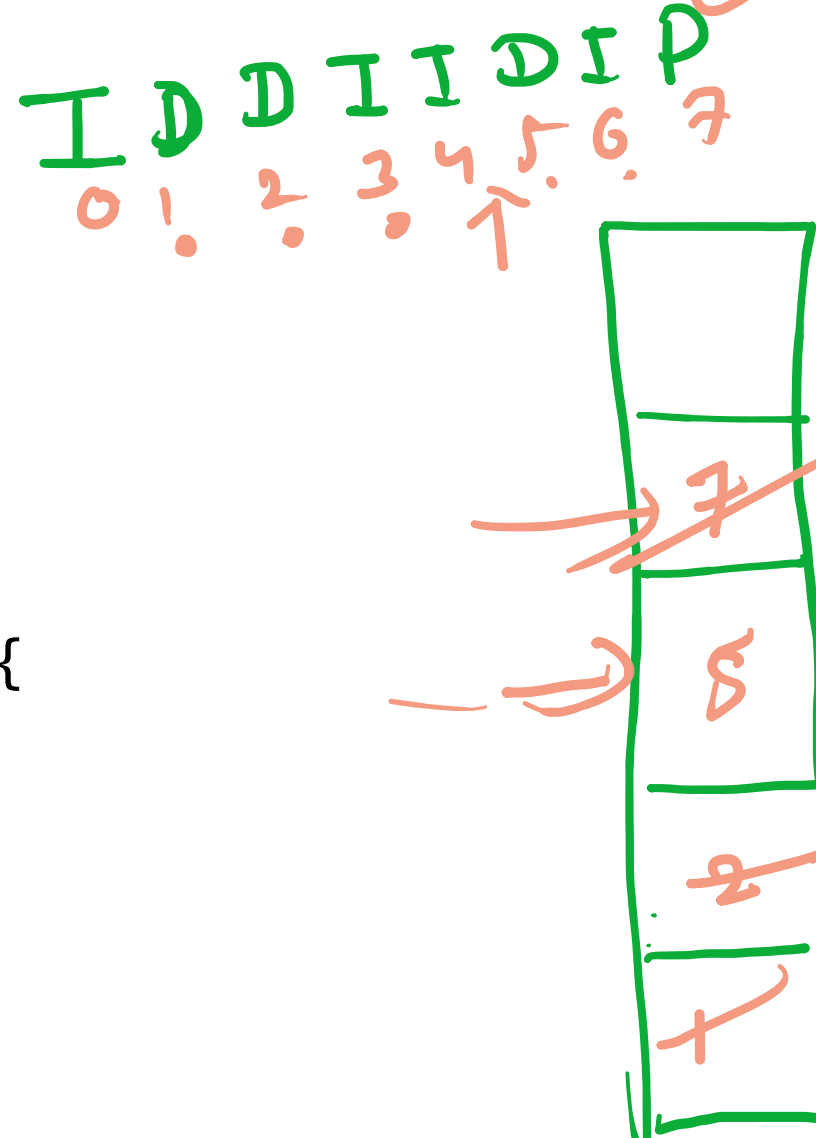
1+0 5
2+0 6
3+0 7
4+0 8
5+0 9

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

Count=2
1 4 3 2 5 7 8 9



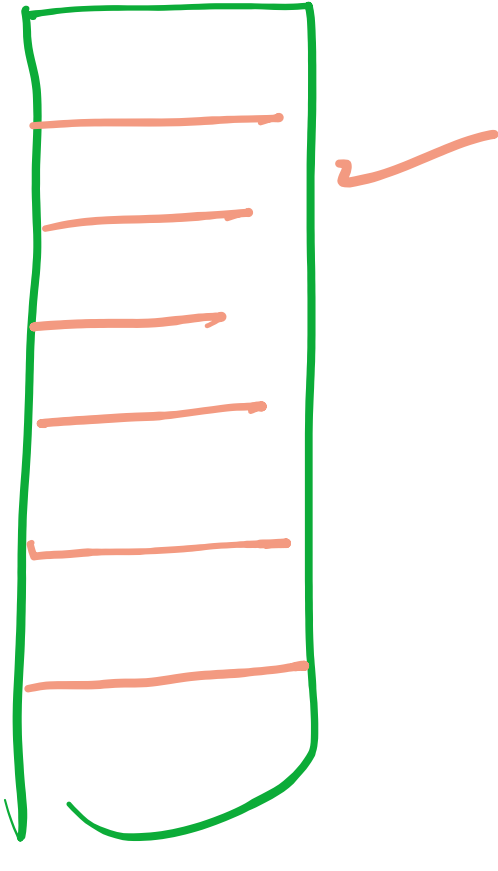
```
public static String smallest_Number(String pattern) {  
    int[] ans = new int[pattern.length() + 1];  
    int count = 1;  
    Stack<Integer> st = new Stack<>();  
    for (int i = 0; i <= pattern.length(); i++) {  
        if (i == pattern.length() || pattern.charAt(i) == 'I') {  
         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



arr(a)(b)=1 -> a -> b
arr(a)(b)=0 -> a -> b
a=3
b=2
arr(3)(2)=1
arr(2)(1)=0
arr(2)(0)=0

st.empty -> plan

((((((3))))))



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

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

