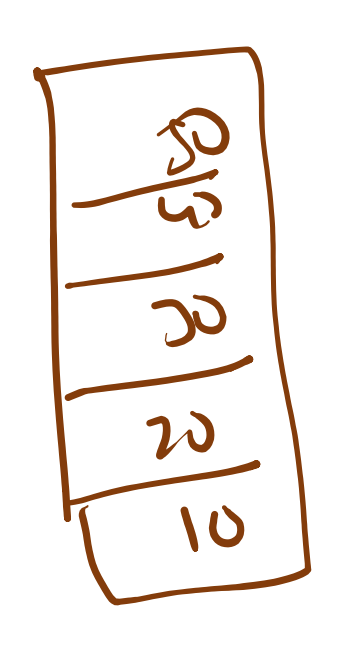
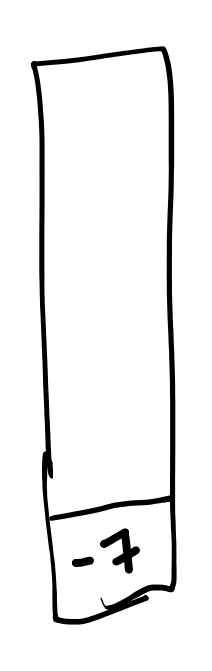
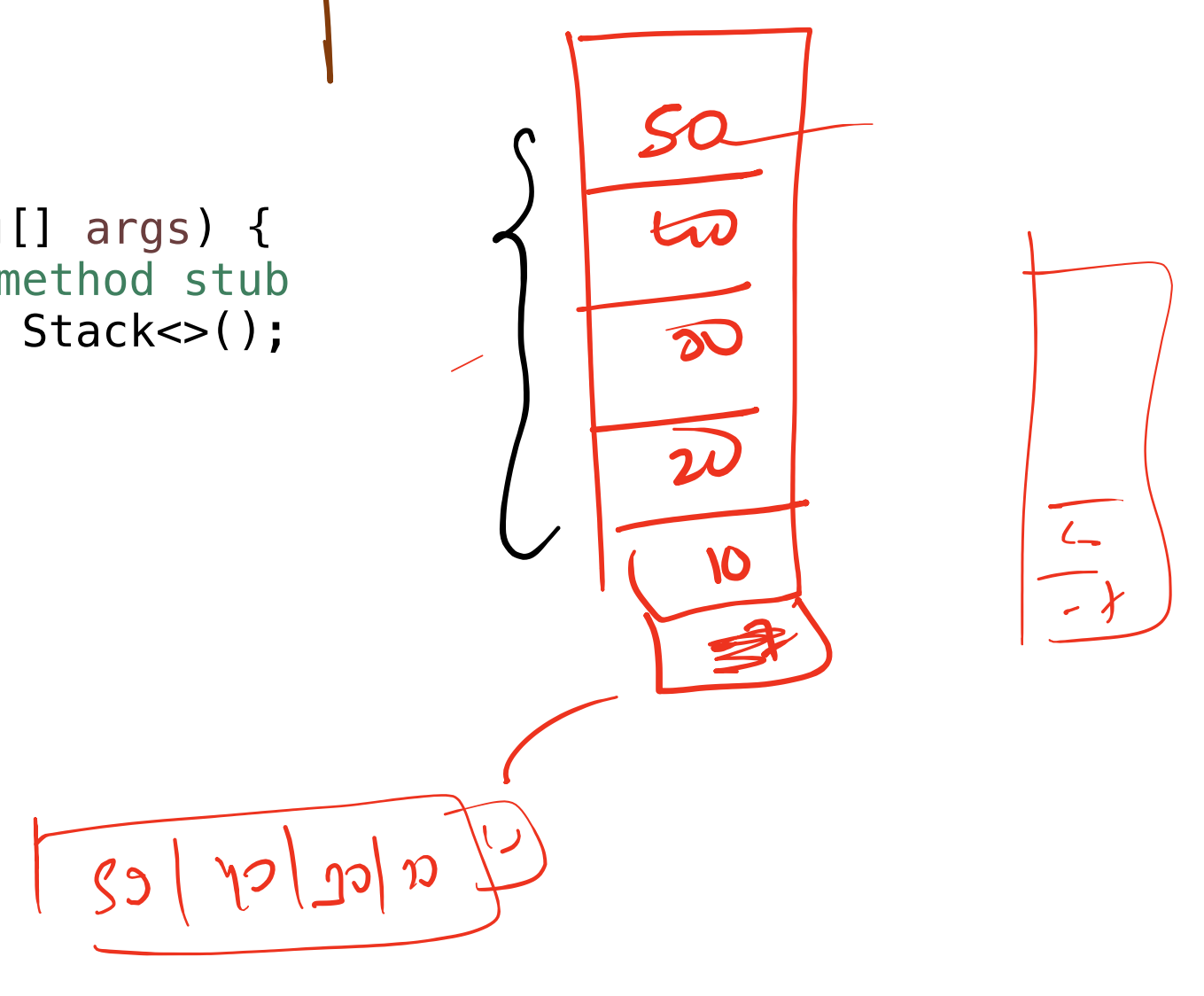
Lec-25

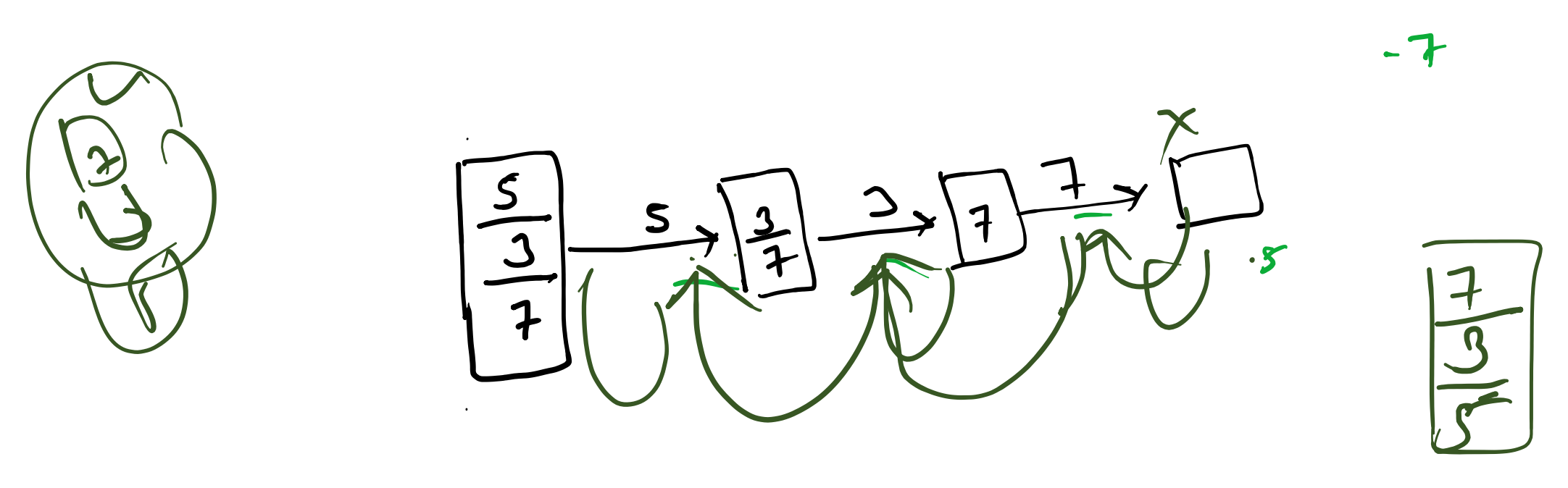
10/08/24 4:45 PM





```java
public static void main(String[] args) {
	// TODO Auto-generated method stub
	Stack<Integer> st = new Stack<>();
	st.push(10);
	st.push(20);
	st.push(30);
	st.push(40);
	st.push(50);
	System.out.println(st);
}
```

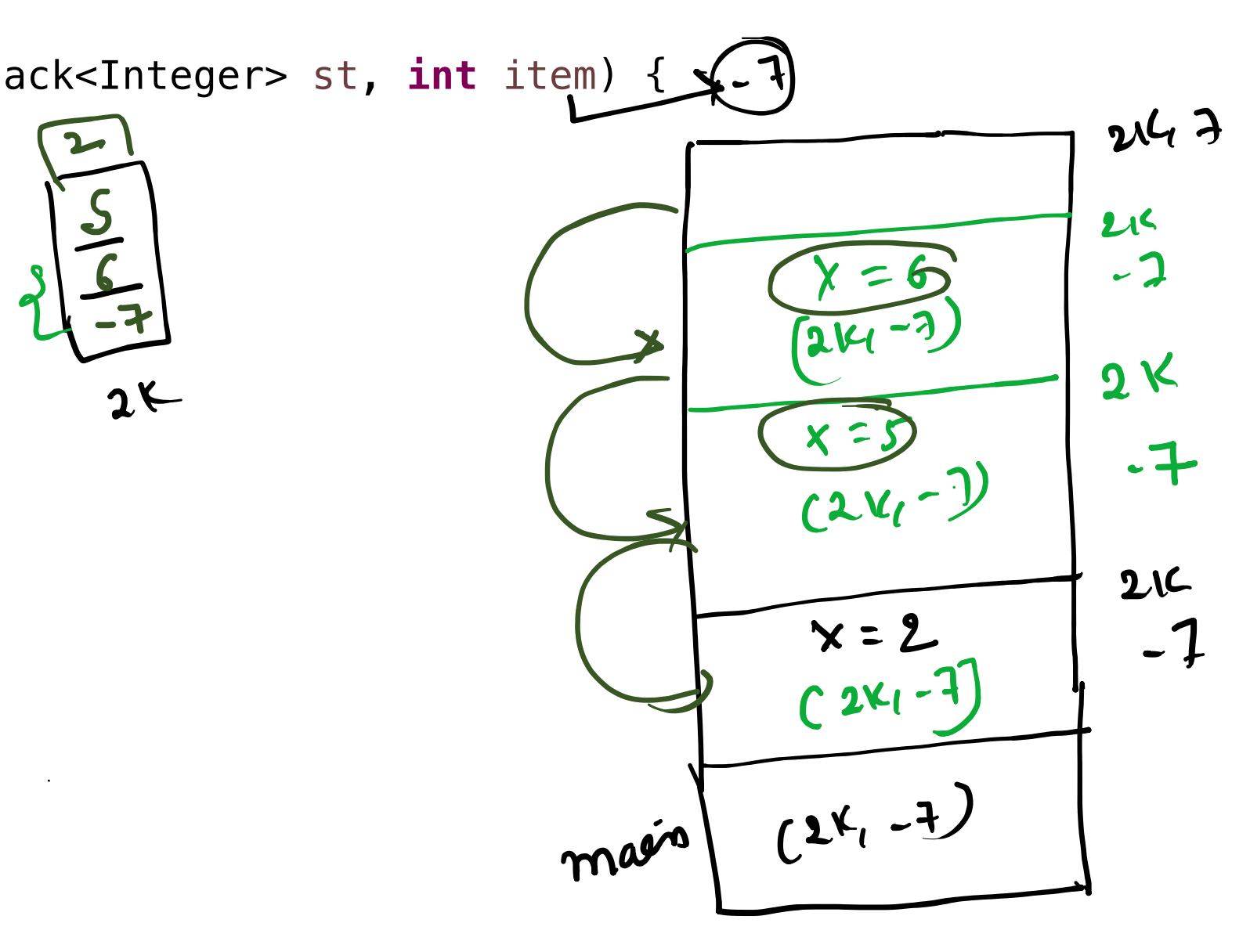




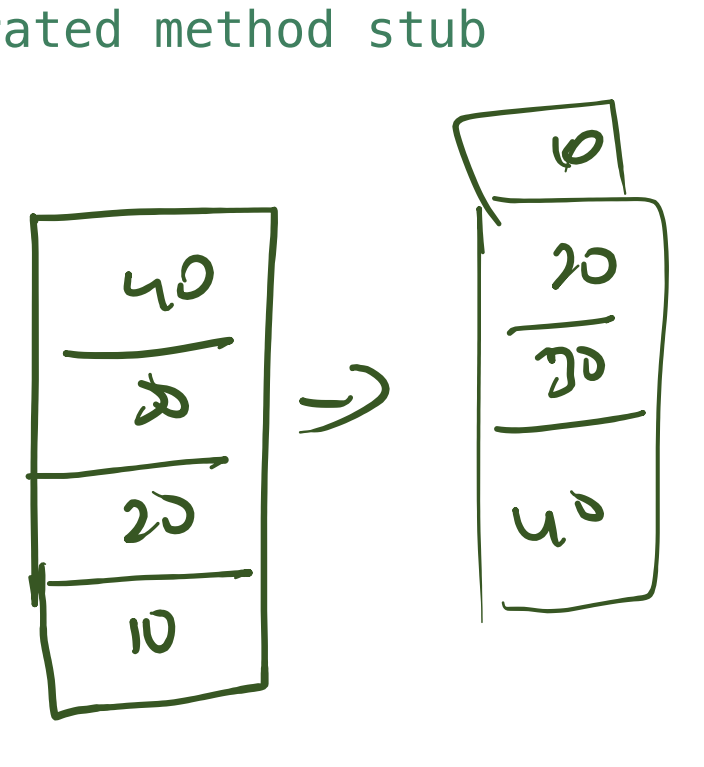
```java
public static void Insert(Stack<Integer> st, int item) {
	if (st.isEmpty()) {
		st.push(item);
		return;
	}
	int x = st.pop();
	Insert(st, item);
	st.push(x);

}
```

-7

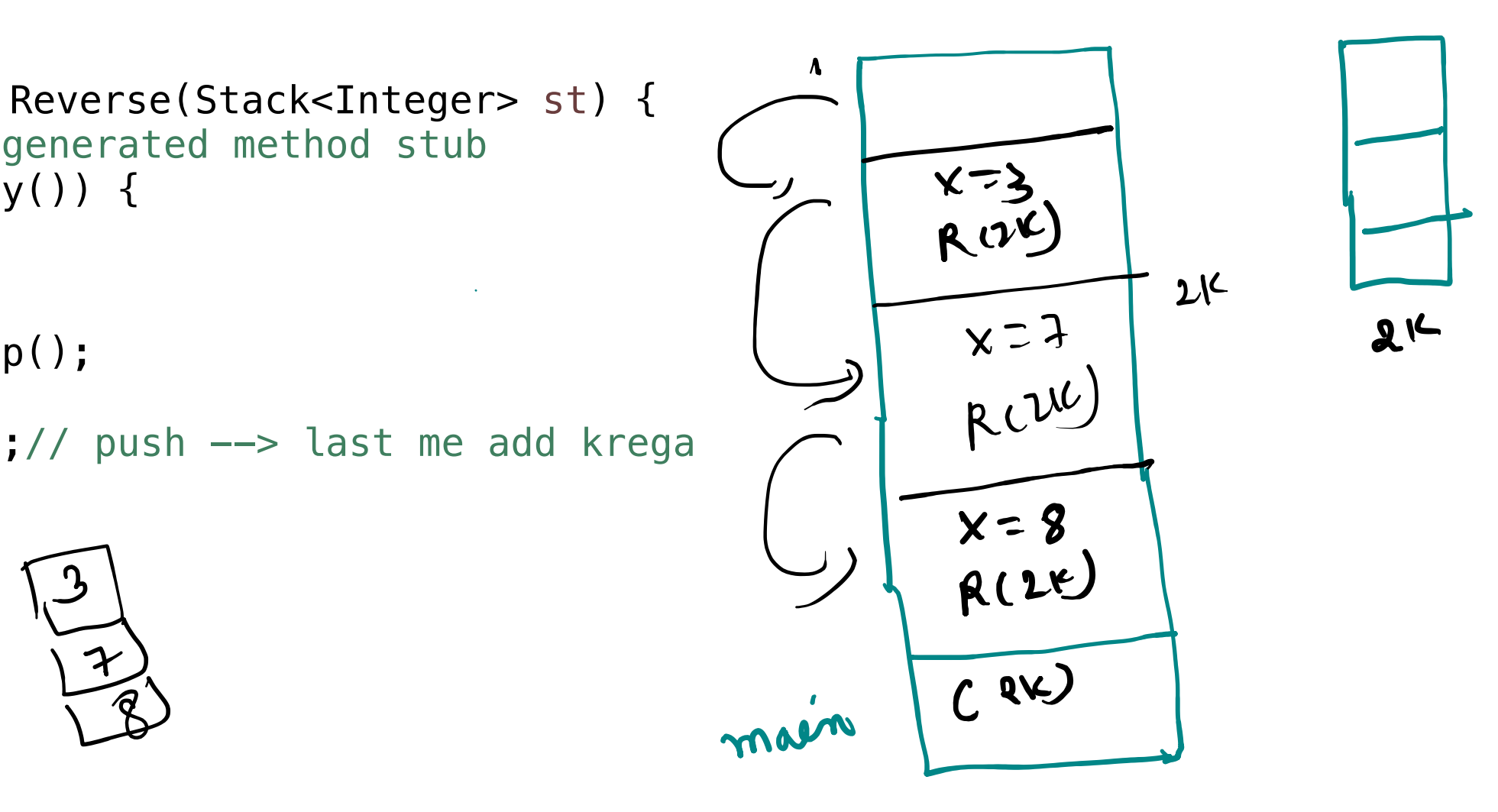
α n)

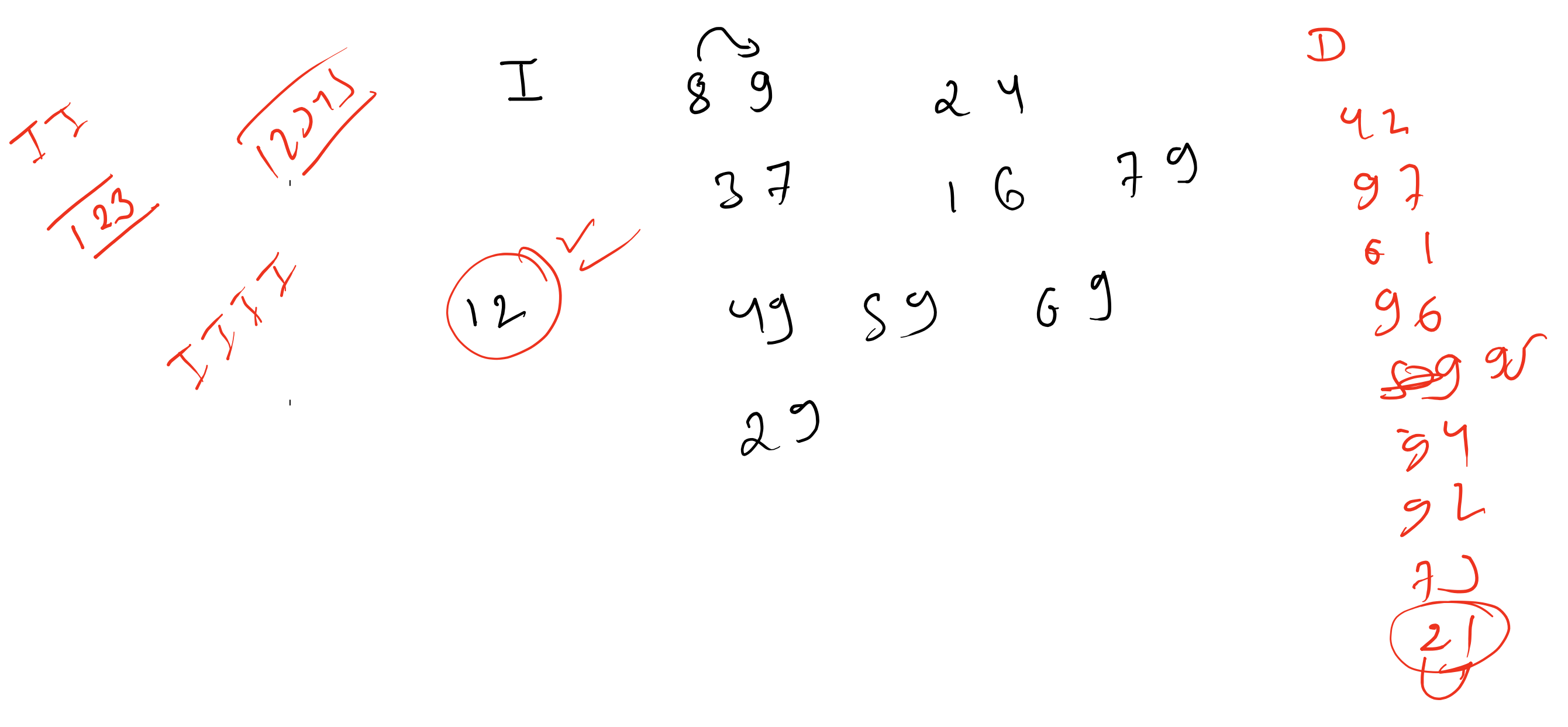


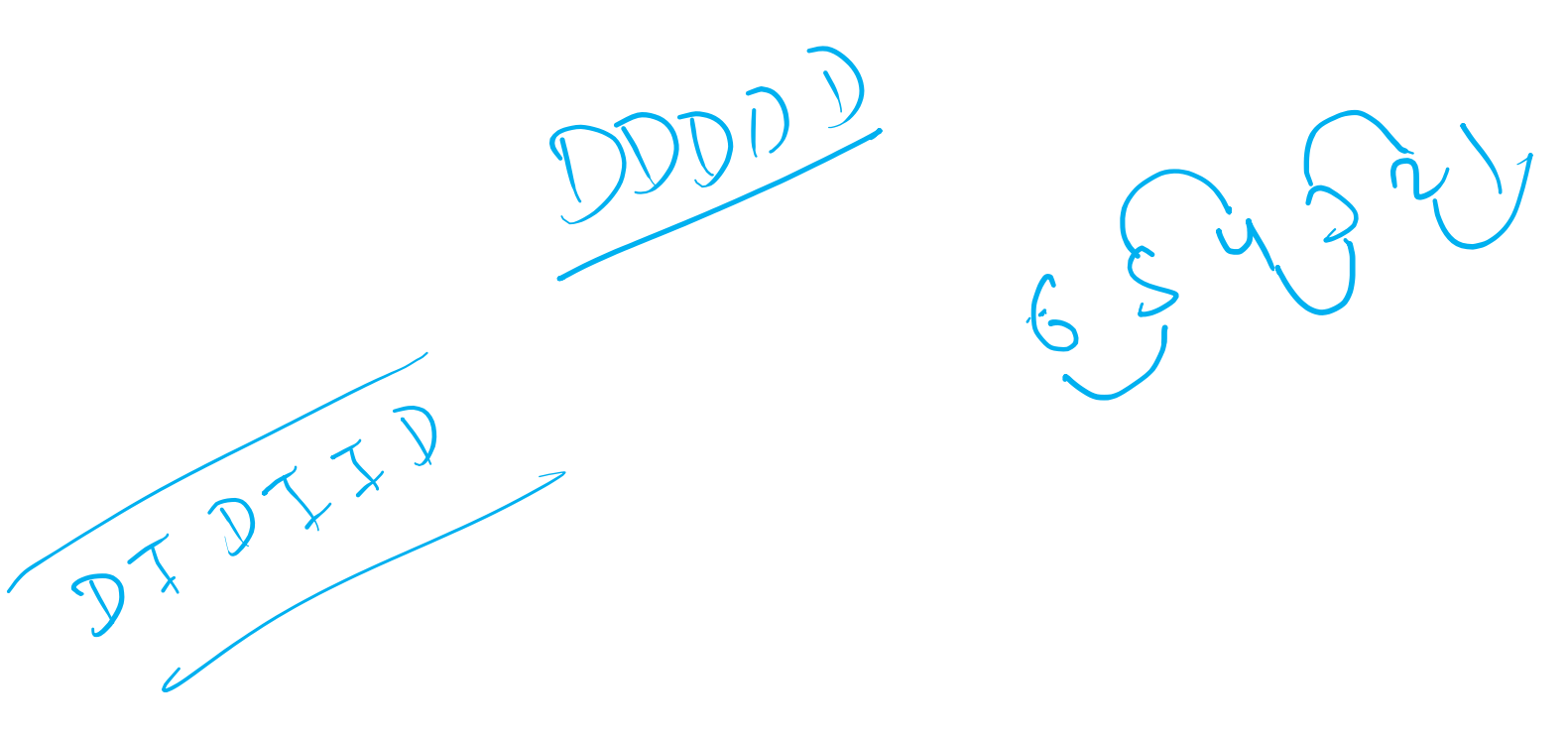
```java
private static void Reverse(Stack<Integer> st) {
	// TODO Auto-generated method stub

}
```

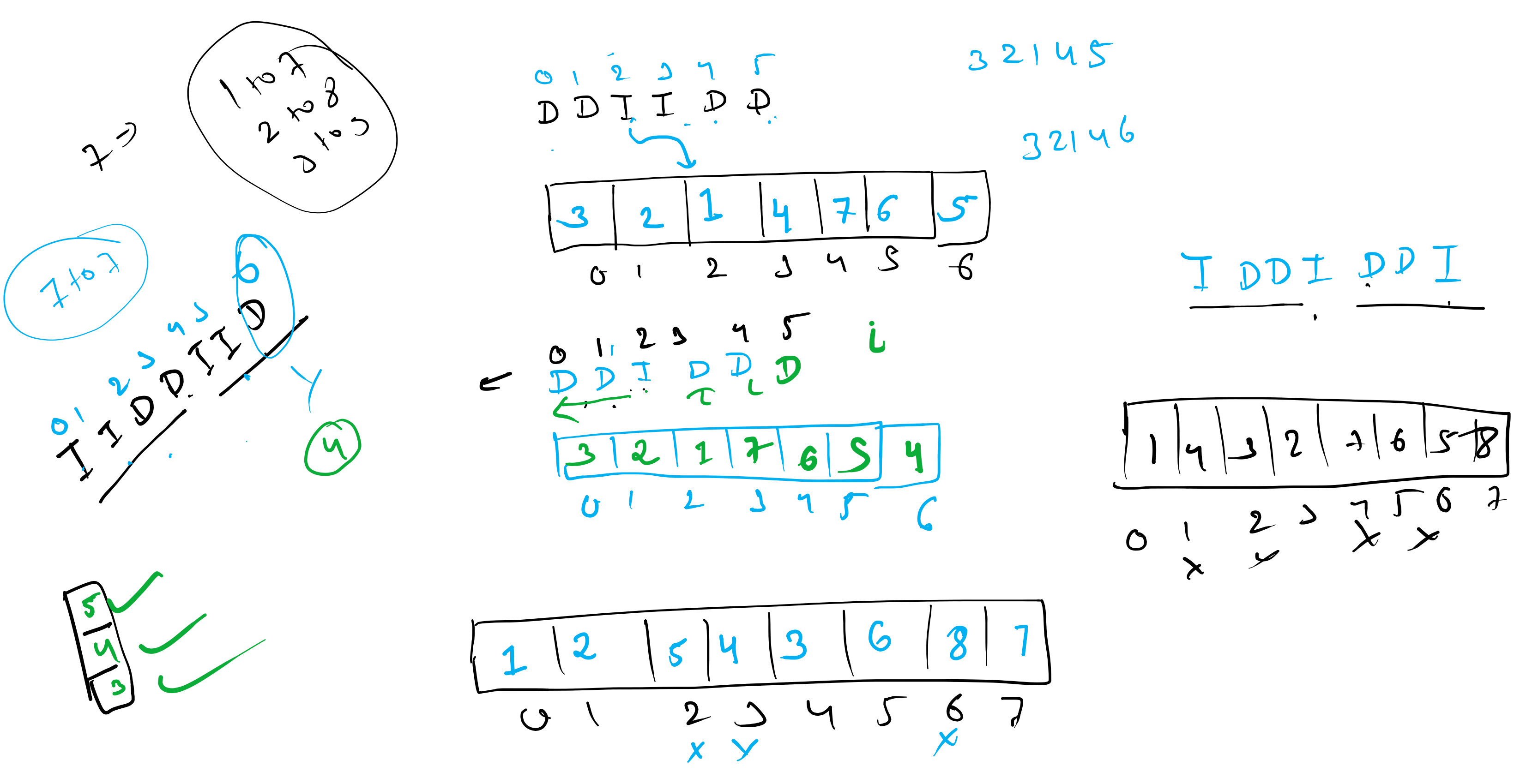


```java
private static void Reverse(Stack<Integer> st) {
	// TODO Auto-generated method stub
	if (st.isEmpty()) {

		return;
	}
	int x = st.pop();
	Reverse(st);
	Insert(st, x);// push --> last me add krega
}
```

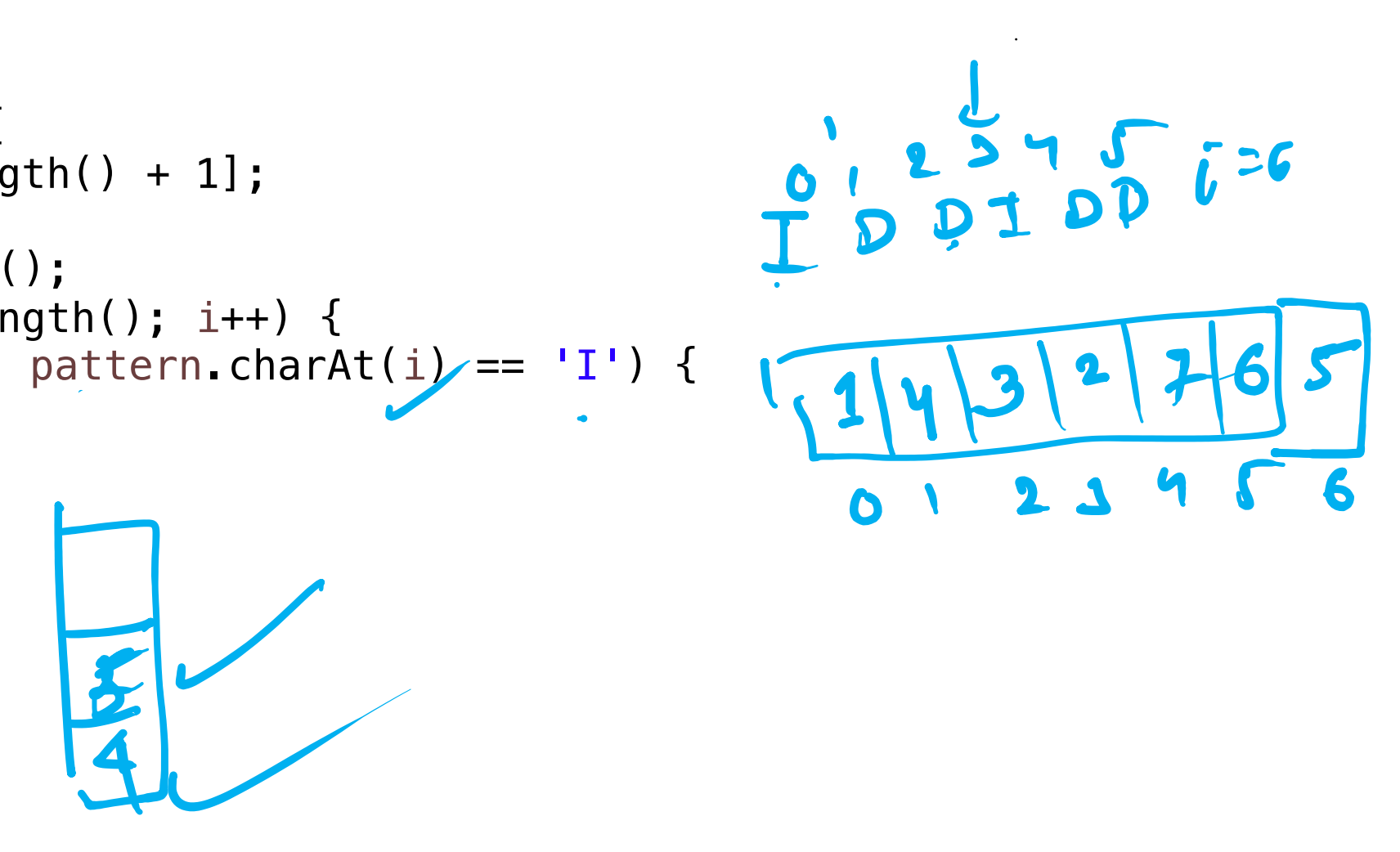


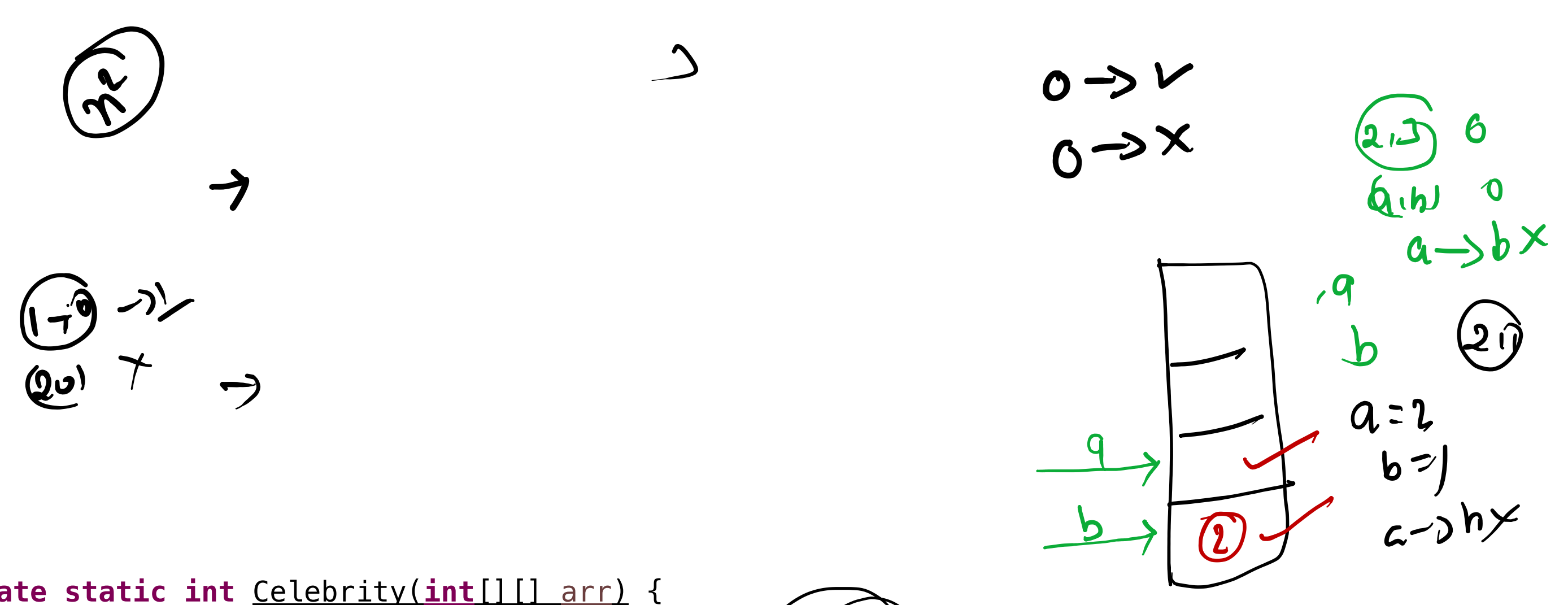




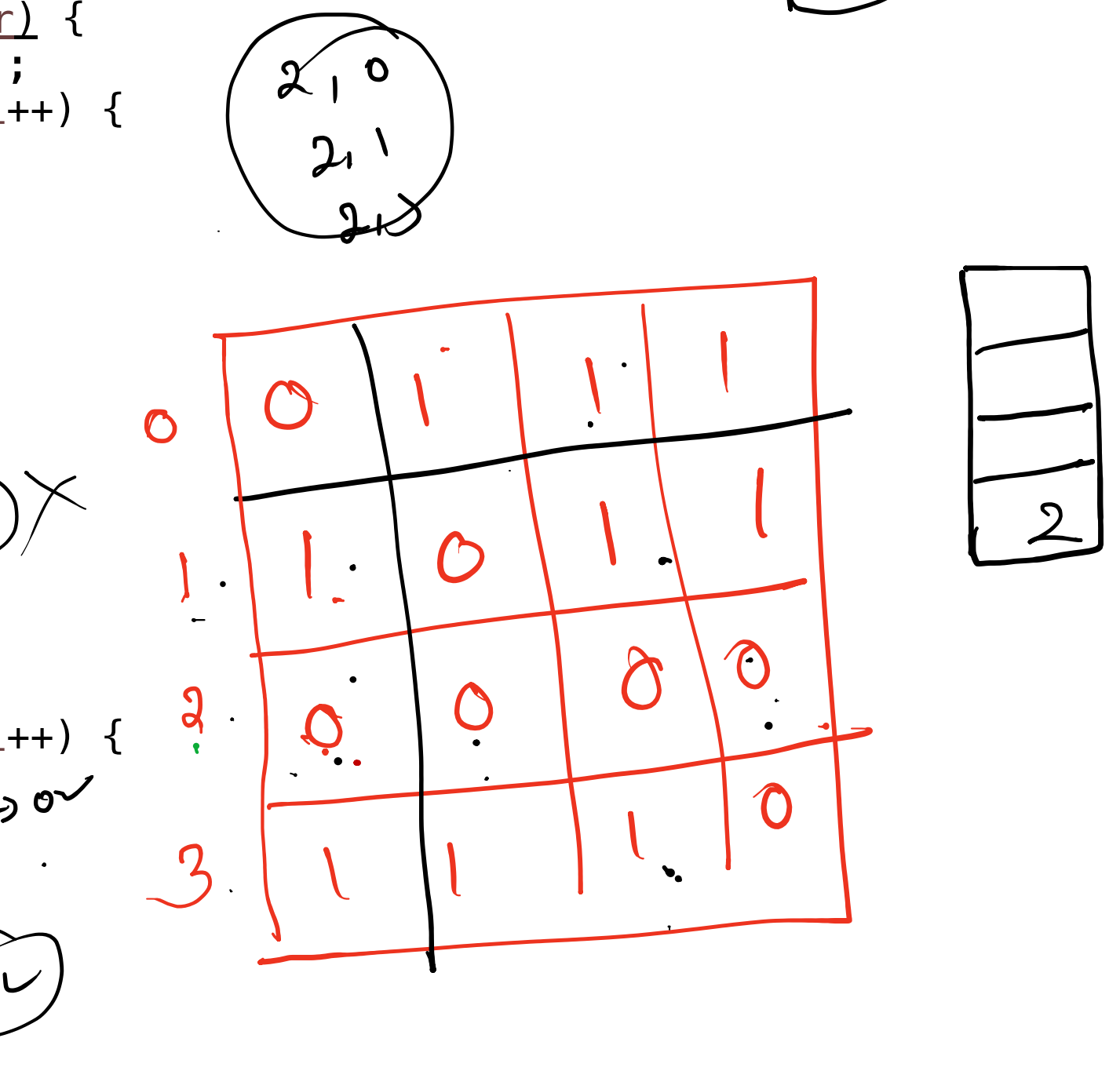


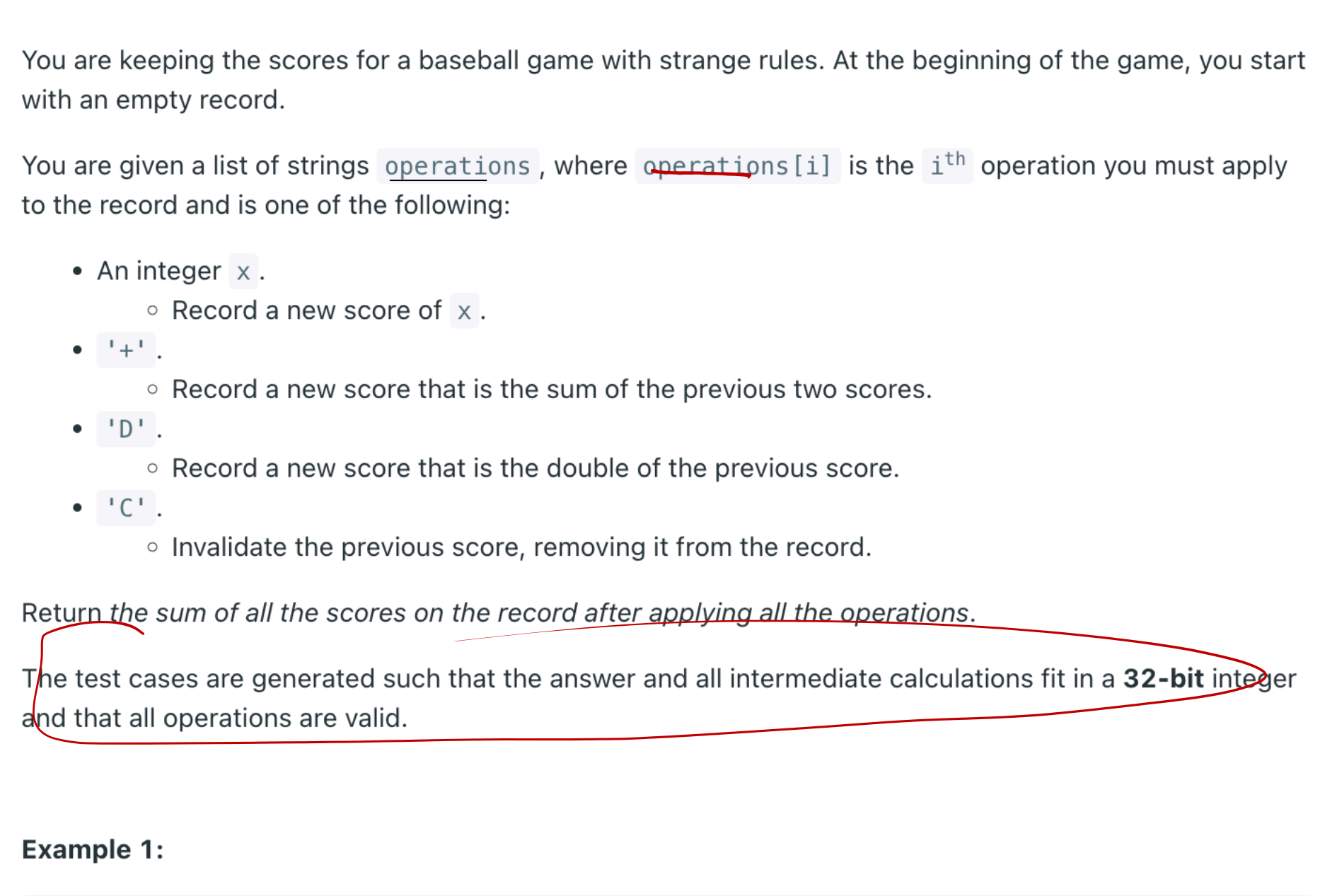
```java
public String Number(String pattern) {
	int[] ans = new int[pattern.length() + 1];
	int c = 1;
	Stack<Integer> st = new Stack<>();
	for (int i = 0; i <= pattern.length(); i++) {
		if (i == pattern.length() || pattern.charAt(i) == 'I') {
			ans[i] = c++;
			// c++;
			while (!st.isEmpty()) {
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

}
```





```java
private static int Celebrity(int[][] arr) {
	Stack<Integer> st = new Stack<>();
	for (int i = 0; i < arr.length; i++) {
		st.push(i);

	}
	while (st.size() > 1) {
		int a = st.pop();
		int b = st.pop();
		if (arr[a][b] == 1) {
			st.push(b);
		} else {
			st.push(a);
		}
	}
	int candidate = st.pop();
	for (int i = 0; i < arr.length; i++) {
		if (i == candidate) {
			continue;
		}
		if (arr[candidate][i] == 1) {
			return -1;
		}
	}

}
```





You are keeping the scores for a baseball game with strange rules. At the beginning of the game, you start with an empty record.

You are given a list of strings `operations`, where `operations[i]` is the `i^th` operation you must apply to the record and is one of the following:

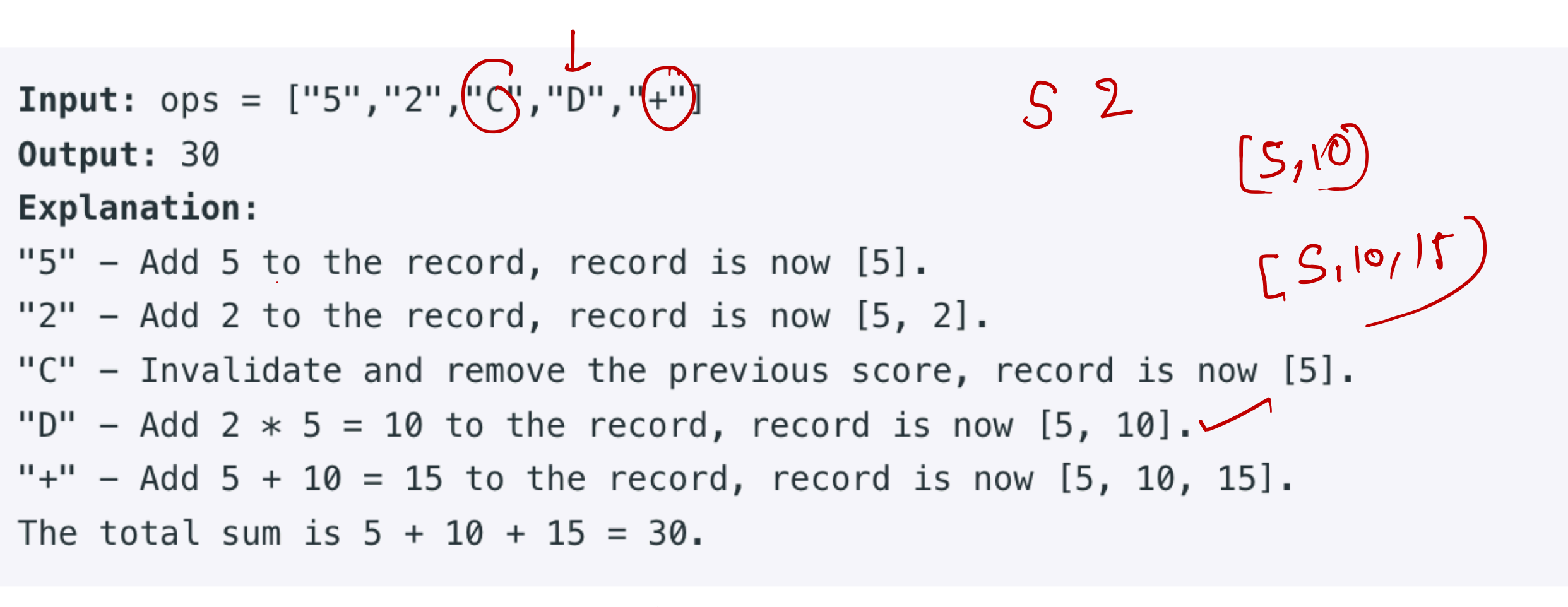
- An integer `x`.
    - Record a new score of `x`.
- `'+'`.
    - Record a new score that is the sum of the previous two scores.
- `'D'`.
    - Record a new score that is the double of the previous score.
- `'C'`.
    - Invalidate the previous score, removing it from the record.

Return *the sum of all the scores on the record after applying all the operations.*

The test cases are generated such that the answer and all intermediate calculations fit in a **32-bit** integer and that all operations are valid.

Example 1:

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

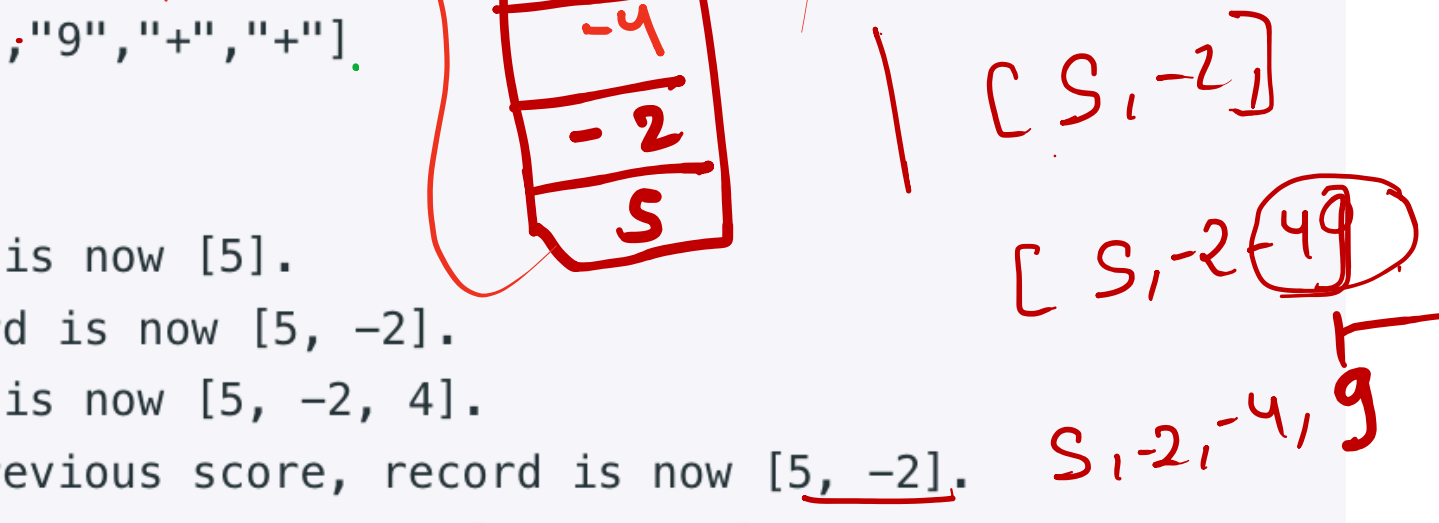


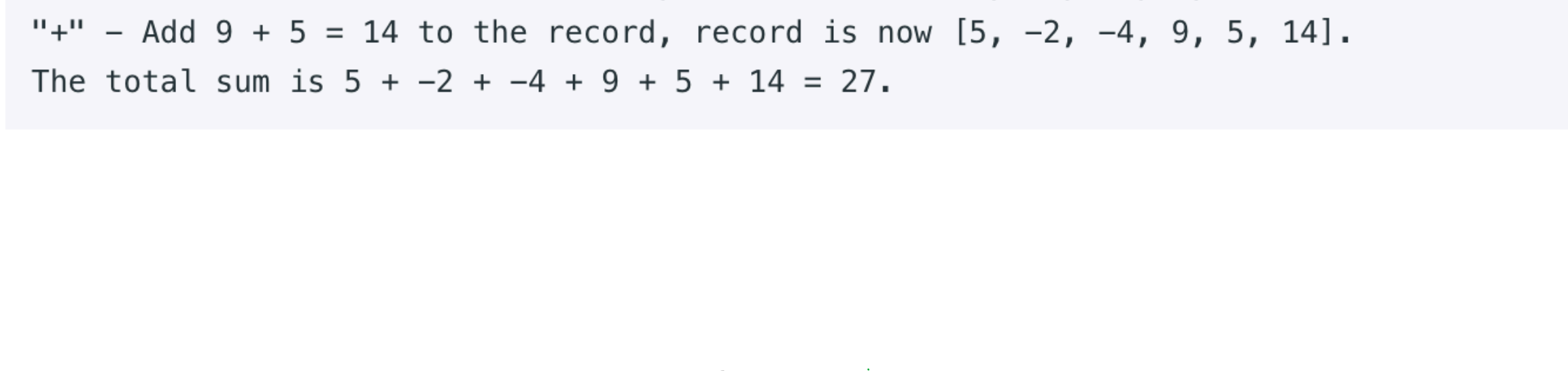
```
Input: ops = ["5","2","C","D","+"]
Output: 30
Explanation:
"5" - Add 5 to the record, record is now [5].
"2" - Add 2 to the record, record is now [5, 2].
"C" - Invalidate and remove the previous score, record is now [5].
"D" - Add 2 * 5 = 10 to the record, record is now [5, 10].
"+" - Add 5 + 10 = 15 to the record, record is now [5, 10, 15].
The total sum is 5 + 10 + 15 = 30.
```

5 2

[5,10]

[5,10,15]





```
Input: ops = ["5","-2","4","C","D","9","+","+"]
Output: 27
Explanation:
"5" - Add 5 to the record, record is now [5].
"-2" - Add -2 to the record, record is now [5, -2].
"4" - Add 4 to the record, record is now [5, -2, 4].
"C" - Invalidate and remove the previous score, record is now [5, -2].
"D" - Add 2 * -2 = -4 to the record, record is now [5, -2, -4].
"9" - Add 9 to the record, record is now [5, -2, -4, 9].
"+" - Add -4 + 9 = 5 to the record, record is now [5, -2, -4, 9, 5].
"+" - Add 9 + 5 = 14 to the record, record is now [5, -2, -4, 9, 5, 14].
The total sum is 5 + -2 + -4 + 9 + 5 + 14 = 27.
```

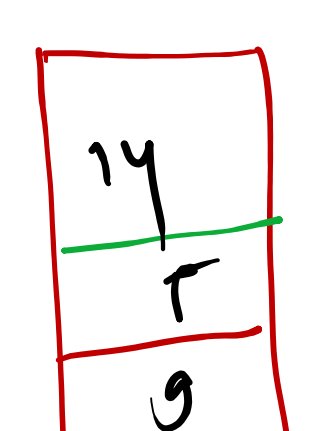
[5,-2]

[5,-2,4,9]

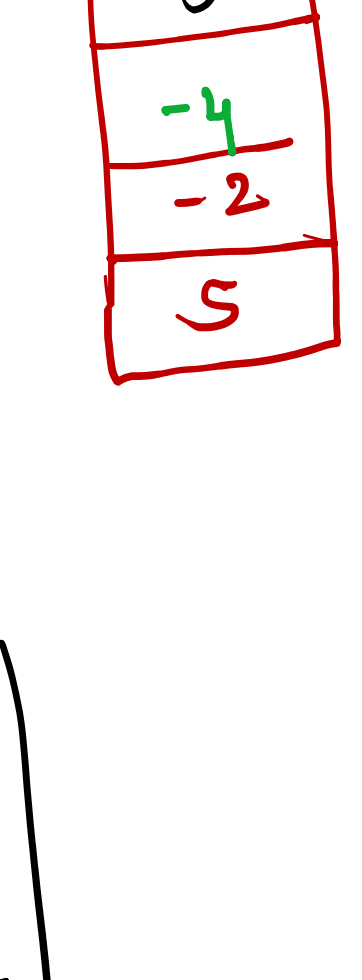
5,-2,-4,9 5

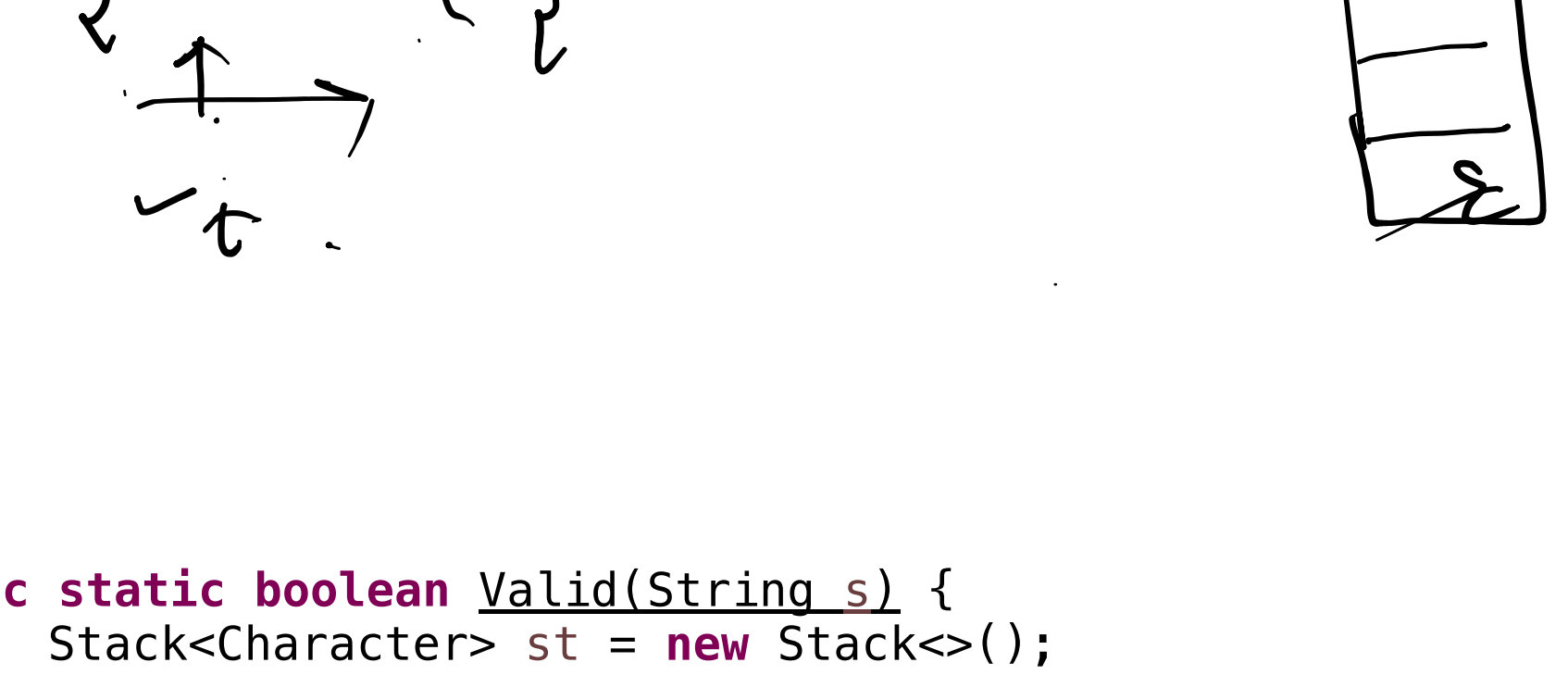
5 -2 -4 9 5 14

```java
public static int Points(String[] a) {
	int sum = 0;
	Stack<Integer> st = new Stack<>();
	for (int i = 0; i < a.length; i++) {
		if (a[i].equals("C")) {
			st.pop();
		} else if (a[i].equals("D")) {
			int x = st.peek();
			st.push(2 * x);
		}else if (a[i].equals("+")) {
			int x = st.pop();
			int y = st.pop();
			st.push(y);
			st.push(x);
			st.push(x + y);
		} else {
			st.push(Integer.parseInt(a[i]));
		}
	}

}
```



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





```java
public static boolean Valid(String s) {
	Stack<Character> st = new Stack<>();
	for (int i = 0; i < s.length(); i++) {
		char ch = s.charAt(i);
		if (ch == '(' || ch == '[' || ch == '{') {
			st.push(ch);
		} else {
			if (st.isEmpty()) {
				return false;
			} else {
				if (ch == ')') {
					if (st.peek() != '(') {
						return false;
					}
					st.pop();

				} else if (ch == '}') {
					if (st.peek() != '{') {
						return false;
					}
					st.pop();

				} else {
					if (st.peek() != '[') {
						return false;
					}
					st.pop();
				}
			}
		}
	}
}
```

"[()]{}"