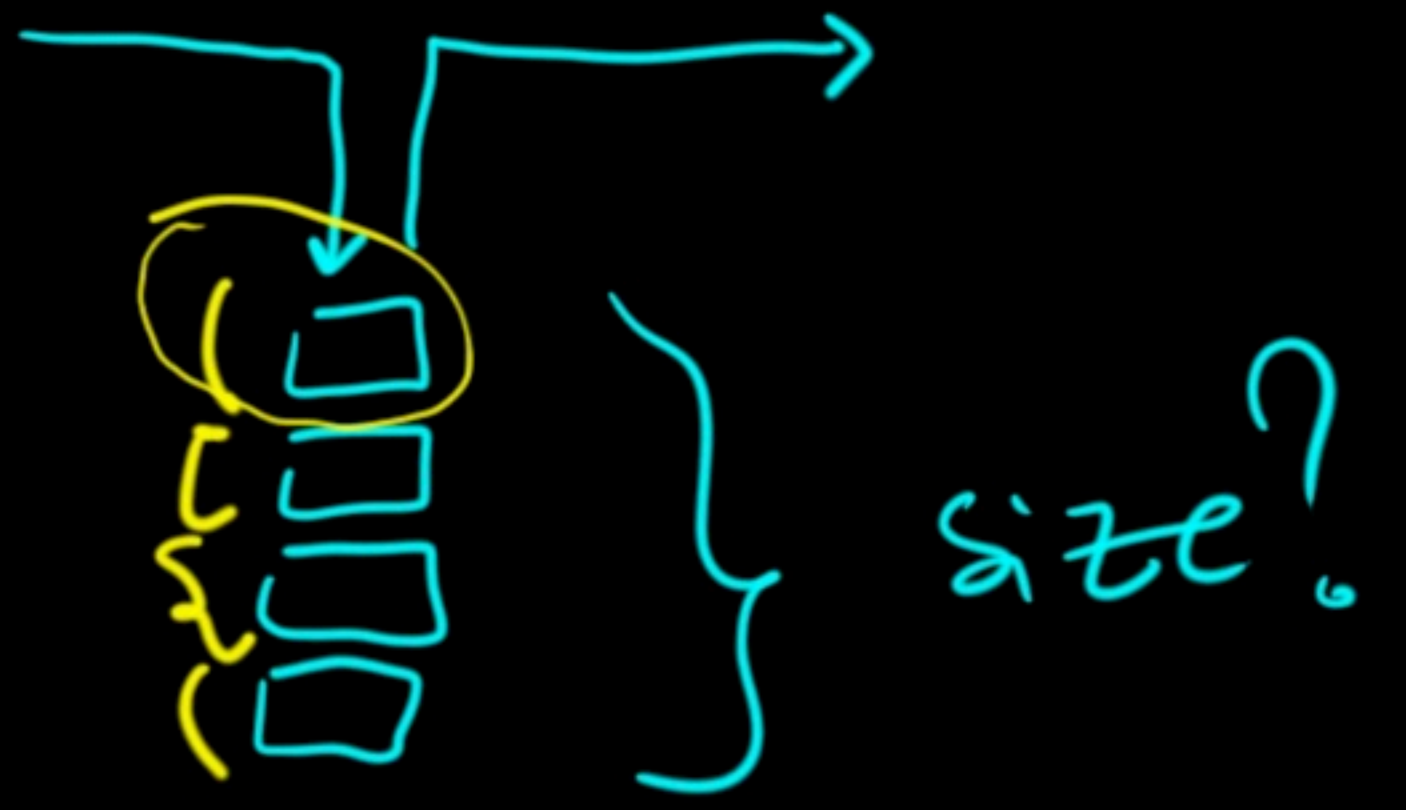
7 2 1 1 1 2 6 5 4 3 2 1 2 2 3 4 5 6 7
?
1!

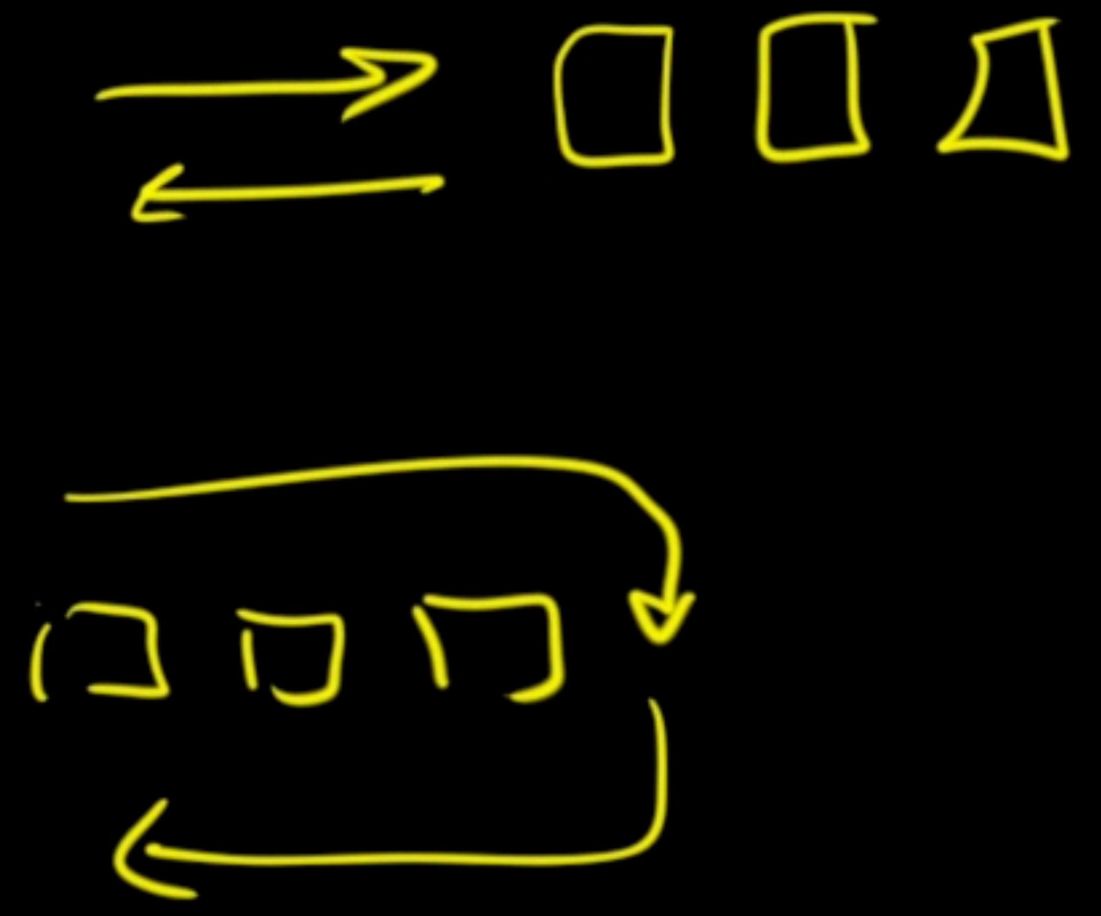
```
{{()[]<>}{[({{()}[]})]}}
```

1.000.000

stack

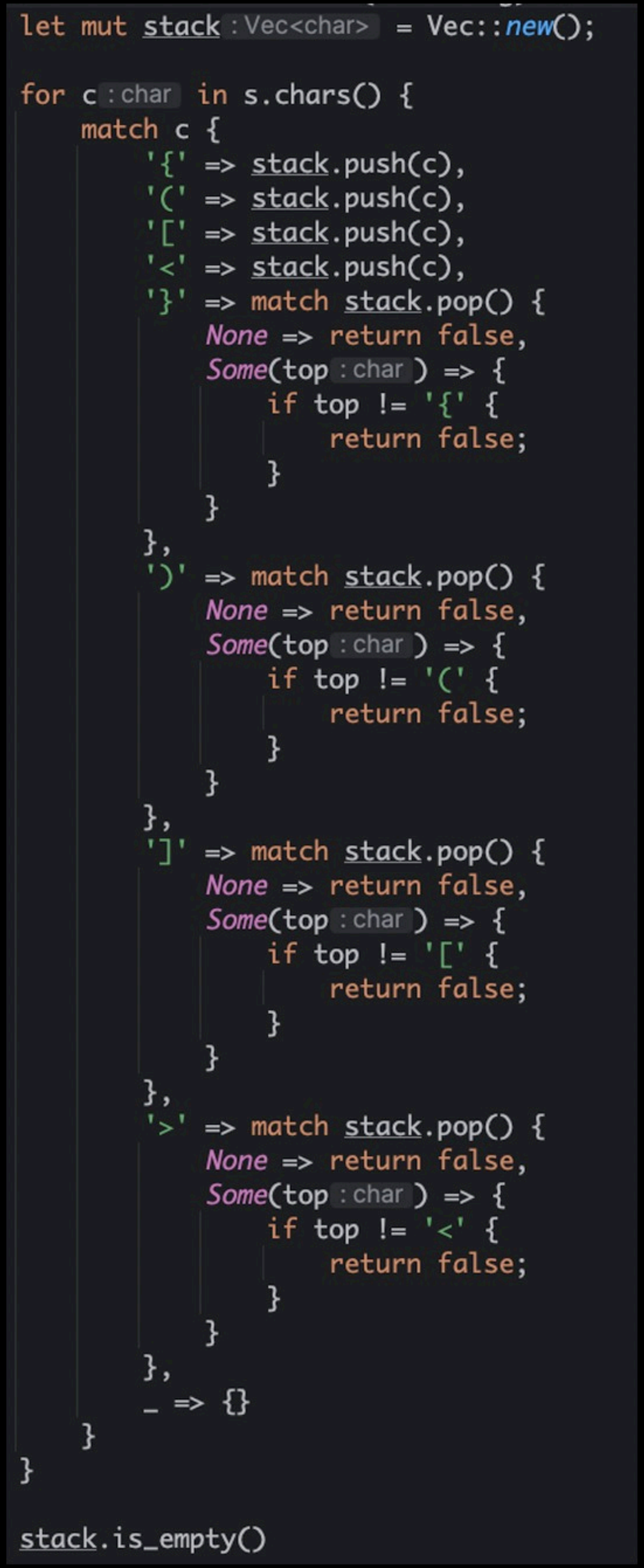


push
pop
size
is_empty



Map
dictionary

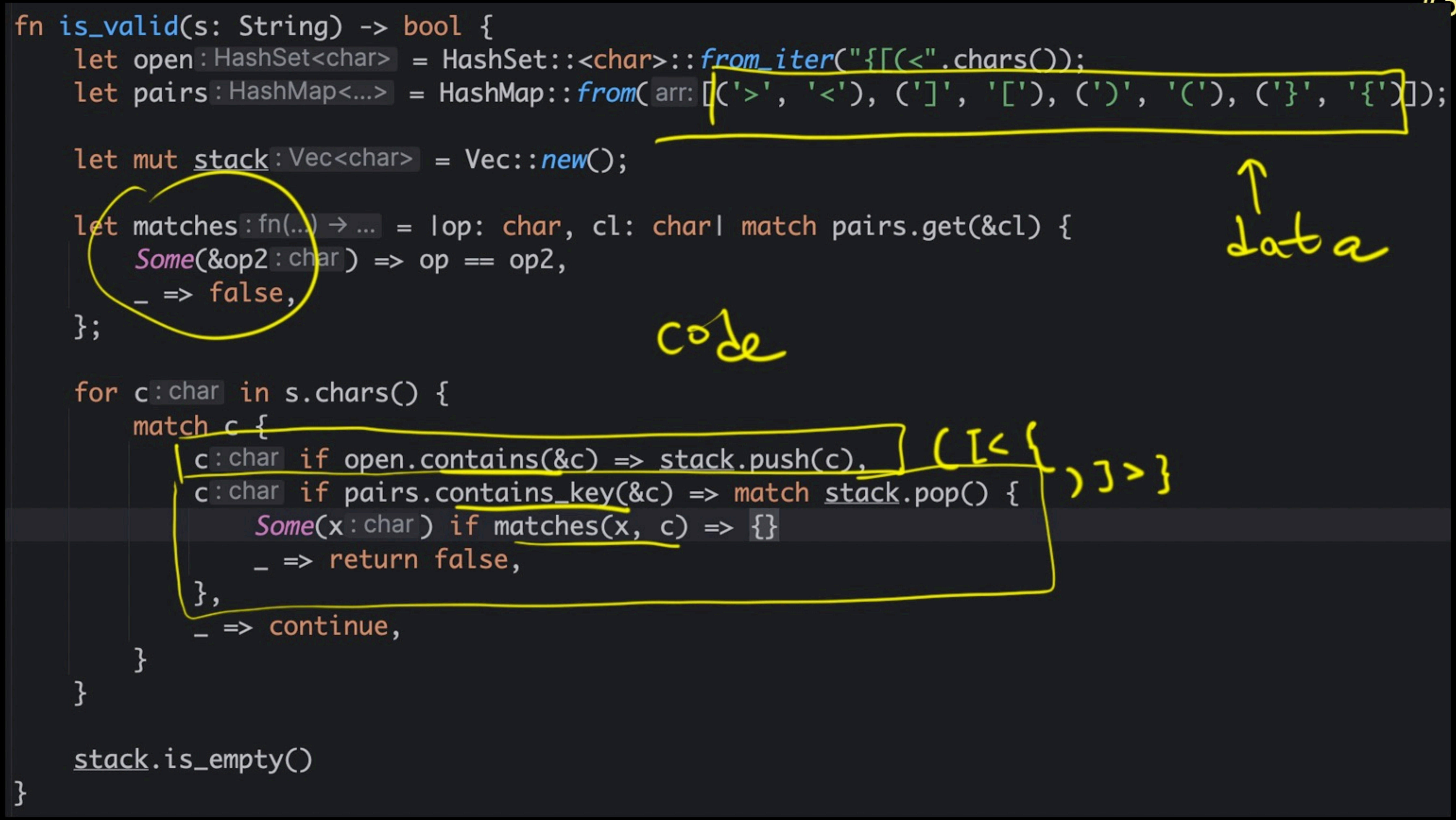
key	value
)	(
}	{
]	[
>	<



```
let mut stack : Vec<char> = Vec::new();

for c : char in s.chars() {
    match c {
        '{' => stack.push(c),
        '(' => stack.push(c),
        '[' => stack.push(c),
        '<' => stack.push(c),
        '}' => match stack.pop() {
            None => return false,
            Some(top : char) => {
                if top != '{' {
                    return false;
                }
            }
        },
        ')' => match stack.pop() {
            None => return false,
            Some(top : char) => {
                if top != '(' {
                    return false;
                }
            }
        },
        ']' => match stack.pop() {
            None => return false,
            Some(top : char) => {
                if top != '[' {
                    return false;
                }
            }
        },
        '>' => match stack.pop() {
            None => return false,
            Some(top : char) => {
                if top != '<' {
                    return false;
                }
            }
        },
        _ => {}
    }
}

stack.is_empty()
```



```
fn is_valid(s: String) -> bool {
    let open : HashSet<char> = HashSet::<char>::from_iter("{[(<".chars());
    let pairs : HashMap<...> = HashMap::from( arr: [('>', '<'), (']', '['), (')', '('), ('}', '{')]);

    let mut stack : Vec<char> = Vec::new();

    let matches : fn(..) -> ... = |op: char, cl: char| match pairs.get(&cl) {
        Some(&op2 : char) => op == op2,
        _ => false,
    };

    for c : char in s.chars() {
        match c {
            c : char if open.contains(&c) => stack.push(c),
            c : char if pairs.contains_key(&c) => match stack.pop() {
                Some(x : char) if matches(x, c) => {}
                _ => return false,
            },
            _ => continue,
        }
    }

    stack.is_empty()
}
```