20. Valid Parentheses (Easy)

You are given a string `s` consisting of the following characters: `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`.

The input string `s` is valid if and only if:

Every open bracket is closed by the same type of close bracket.Open brackets are closed in the correct order.Every close bracket has a corresponding open bracket of the same type.

- Every open bracket is closed by the same type of close bracket.
- Open brackets are closed in the correct order.
- Every close bracket has a corresponding open bracket of the same type.

Return `true` if `s` is a valid string, and `false` otherwise.

Example 1:

```
Input: s = "[]"

Output: true
```

Example 2:

```
Input: s = "([{}])"

Output: true
```

Example 3:

```
Input: s = "[(])"

Output: false
```

Explanation: The brackets are not closed in the correct order.

Constraints:

- `1 <= s.length <= 1000`

```
class Solution {
public:
    bool isValid(string s) {
        stack<char> stack;
        pair<char,char> a= {'(',')'},b= {'[',']'},c= {'{','}'};
        char tmp;
        for(char cc:s){
            switch (cc){
                case ')':
                if(stack.empty() || stack.top() != a.first) return false;
                stack.pop();
                break;
                case ']':
                if(stack.empty() || stack.top() != b.first) return false;
                stack.pop();
                break;
                case '}':
                if(stack.empty() || stack.top() != c.first) return false;
                stack.pop();
                break;
                default: stack.push(cc);
            }
        }
        return stack.empty();
    }
};
```