

Office Hours #3

```
1 #include <stack>
2
3 bool isPalindrome(string &s) {
4     stack<char> mStack;
5
6     string temp;
7     for (int i = 0; i < s.length(); i++) {
8         mStack.push(s[i])
9     }
10    for (int i = 0; i < s.length(); i++) {
11        temp += mStack.top();
12        mStack.pop();
13    }
14
15    if (temp == s) return true;
16    else return false;
17 }
```

```
1 #include <stack>
2
3 bool matchingBrackets(string &s) {
4     stack<char> ms;
5     for (int i = 0; i < s.length(); i++) {
6         switch (s[i]) {
7             case '(':
8             case '[':
9             case '{':
10                ms.push(s[i]);
11                break;
12             case ')':
13                if (ms.top() == '(')
14                    ms.pop();
15                else
16                    return false;
17                break;
18             case ']':
19                if (ms.top() == '[')
20                    ms.pop();
21                else
22                    return false;
23                break;
24             case '}':
25                if (ms.top() == '{')
26                    ms.pop();
27                else
28                    return false;
29                break;
30        }
31    }
32
33    if (!ms.empty()) return false;
```

```
34     else return true;
35 }
```

```
1 // Diary Class Challenge from Lecture 7
```

```
2
3 #include <iostream>
4 #include <string>
5
6 class Diary {
7     public:
8         Diary(string name) {
9             m_name = name;
10        }
11
12        virtual ~Diary() {
13
14        }
15
16        string getTitle() const {
17            return m_name;
18        }
19
20        virtual void writeEntry(string entry) {
21            m_entries += entry;
22        }
23
24        virtual string read() const {
25            return m_entries;
26        }
27
28    private:
29        string m_name;
30        string m_entries;
31 };
32
33 class SecretDiary: public Diary {
34     public:
35
36         SecretDiary(string name): Diary("TOP-SECRET") {
37
38        }
39
40        virtual ~SecretDiary(){
41
42        }
43
44        virtual void writeEntry(string entry) {
45            Diary::writeEntry(encode(s));           // don't touch Diary's privates!
46        }
47
48        virtual string read() {
49            return decode(Diary::read());           // don't touch Diary's privates!
50        }
51 };
```