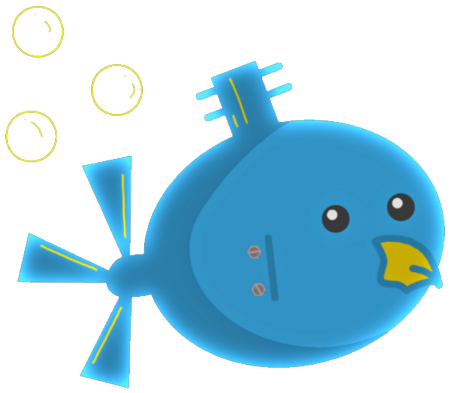
CALIFORNIA STATE UNIVERSITY, LOS ANGELES

Module Level Outcome 2: *Discrete Mathematics and Automata Theory*



ROBOSUB

Members
Thomas BENSON, David CAMACHO, Bailey CANHAM, Brandon CAO,
Roberto HERNANDEZ, Andrew HEUSSER, Hector MORA-SILVA,
Bart RANDO, Victor SOLIS

February 21, 2023

Question 3: Valid Parentheses

Problem: https://leetcode.com/problems/valid-parentheses/

Pseudocode

```
function isValid($s$)                                    ▷ Valid parentheses in string s
    $A \leftarrow []$                                                        ▷ Create Stack
    for $i \leftarrow 0, s.length - 1$ do
        if $s[i] =$ ( or $s[i] =$ [ or $s[i] =$ { then
            $A.push(s[i])$
        else if $s[i] =$) or $s[i] =$] or $s[i] =$} then
            if $A.top \neq$ ( or $A.top \neq$ [ or $A.top \neq$ { then
                $break$
            else
                $A.pop()$
            end if
        end if
    end for
    if $A.empty()$ then
        return true
    else
        return false
    end if
end function
```

Code

C++

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

Java

```
class Solution {
    public boolean isValid(String s) {
        Stack<Character> stack = new Stack<>();
        for (char ch: s.toCharArray()) {
            switch(ch) {
                case('('):
                case('['):
                case('{'):
                    stack.push(ch);
                    break;
                case(')'):
                    if (stack.empty() || stack.peek() != '(') {
                        return false;
                    }
                    stack.pop();
                    break;
                case(']'):
                    if (stack.empty() || stack.peek() != '[') {
                        return false;
                    }
                    stack.pop();
                    break;
                case('}'):
                    if (stack.empty() || stack.peek() != '{') {
                        return false;
                    }
                    stack.pop();
                    break;
            }
        }
        return stack.rmpty();
    }
}
```

Python

```
    class Solution(object):
        def isValid(self, s):
            stack = []
            characters = {"(": ")", "{": "}", "[": "]"}
            for char in s:
                if char in characters:
                    top_element = stack.pop() if stack else '#'
                    if characters[char] != top_element:
                        return False
                    else:
                        stack.append(char)

            return not stack
```

Question 4: Regular Expression Matching

Problem: https://leetcode.com/problems/regular-expression-matching/

Pseudocode

```
function IsMatch(string $s$, string $p$)
    function DFS(int $i$, int $j$)
        if $i \geq s.length$ then
```

```
            if $j \geq p.length$ then
                return True
            end if
        end if
        if $j \geq p.length$ then
            return False
        end if
        $a \leftarrow i < s.length$
        $b \leftarrow s[i] == p[i]$ or $p[j] == .$
        $match \leftarrow a$ and $b$
        if $j+1 < p.length$ and $p[j+1] = *$ then
            return $DFS(i, j+2)$ or $(match$ and $DFS(i+1, j))$
        end if
        if $match$ then
            return $DFS(i+1, j+1)$
        else
            return False
        end if
    end function
end function
```

Code

Python

```
class Solution:
    def isMatch(self, s: str, p: str) -> bool:
        cache = {}

        def dfs(i, j):
            if (i, j) in cache:
                return cache[(i, j)]

            #Base cases
            #if both iterated to end of strings
            if i >= len(s) and j >= len(p):
                return True

            #if j is out of bounds but i is still in bounds
            if j >= len(p):
                return False

            match = i < len(s) and (s[i] == p[j] or p[j] ==".")

            if (j + 1) < len(p) and p[j + 1] == "*":

                #either repeat once or zero times
                cache[(i, j)] = (dfs(i, j + 2) or
                                (match and dfs(i + 1, j)))
                return cache[(i, j)]
            if match:
                cache[(i,j)] = dfs(i + 1, j + 1)
                return cache[(i,j)]
            cache[(i,j)] = False
            return False
        return dfs(0, 0)
```

JavaScript

```
    const isMatch = (s, p) => {
    const dfs = (i, j) => {
        if (i >= s.length && j >= p.length) {
            return true;
        }
        if (j >= p.length) {
            return false;
        }
        const match = i < s.length && (s[i] === p[j] || p[j] === '.');
        if (((j + 1) < p.length) && p[j + 1] === '*') {
            return (dfs(i, j + 2) ||
                    (match && dfs(i + 1, j))
                );
        }
        if (match) {
            return dfs(i + 1, j + 1);
        }
        return false;
    }
    return dfs(0, 0)
}
```