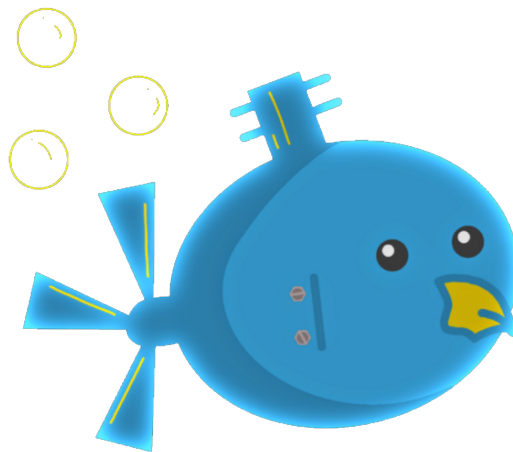


CALIFORNIA STATE UNIVERSITY, LOS ANGELES

---

**Module Level Outcome: *Computer  
Systems/Architectures: Hardware/Operating  
Systems/Networking/Database***

---



ROBOSUB

*Members*

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

March 21, 2023

## Contents

<b>1</b>	<b>Question 7</b>	<b>2</b>
1.1	Pseudocode . . . . .	2
1.2	Code . . . . .	2
1.2.1	Java . . . . .	2
1.2.2	JavaScript . . . . .	3
<b>2</b>	<b>Question 8</b>	<b>4</b>
2.1	Pseudocode . . . . .	4
2.2	Code . . . . .	5
2.2.1	JavaScript . . . . .	5

## 1 Question 7

Problem: <https://leetcode.com/problems/unique-paths/>

### 1.1 Pseudocode

---

**Algorithm 1** Unique Paths

---

```
1: procedure UNIQUEPATHS( $m, n$ )
2:    $hold[n] \leftarrow 1$ 
3:    $count \leftarrow 1$ 
4:   for  $j \leftarrow 0$  to  $n$  do
5:      $hold[j] \leftarrow 1$ 
6:   end for
7:   while  $count < m$  do
8:     for  $i \leftarrow 1$  to  $n$  do
9:        $hold[i] \leftarrow hold[i] + hold[i - 1]$ 
10:    end for
11:     $count \leftarrow count + 1$ 
12:  end while
13:  return  $hold[n - 1]$ 
14: end procedure
```

---

### 1.2 Code

#### 1.2.1 Java

```
class Solution {
    public int uniquePaths(int m, int n) {
        int[] hold = new int[n];
        int count = 1;

        for(int j = 0; j < n; j++){
            hold[j] = 1;
        }

        while(count < m){
            for(int i = 1; i < n; i++){
                hold[i] = hold[i] + hold[i-1];
            }

            count++;
        }

        return hold[n-1];
    }
}
```

### 1.2.2 JavaScript

```
const uniquePaths = (m, n) => {  
  const grid = new Array(m).fill(null).map(() => new Array(n).fill(1));  
  
  for (let row = 1; row < m; row++) {  
    for (let col = 1; col < n; col++) {  
      grid[row][col] = grid[row - 1][col] + grid[row][col - 1];  
    }  
  }  
  
  return grid[m - 1][n - 1];  
}
```

## 2 Question 8

Problem: <https://leetcode.com/problems/word-search/>

### 2.1 Pseudocode

---

**Algorithm 2** Word Search

---

```
1: procedure EXIST(board, word)
2:   if !board or !word then
3:     return false
4:   end if
5:   rows  $\leftarrow$  board.length
6:   cols  $\leftarrow$  board[0].length
7:   procedure DFS(row, col, index)
8:     if row < 0 or row  $\geq$  rows or col < 0 or col  $\geq$  cols or board[row][col]  $\neq$ 
       word[index] then
9:       return false
10:    end if
11:    if index = word.length - 1 then
12:      return true
13:    end if
14:    temp  $\leftarrow$  board[row][col]
15:    board[row][col]  $\leftarrow$  "/"
16:    directions  $\leftarrow$  [[-1, 0], [1, 0], [0, -1], [0, 1]]
17:    for [dx, dy]  $\leftarrow$  directions do
18:      if dfs(row + dx, col + dy, index + 1) then
19:        return true
20:      end if
21:    end for
22:    board[row][col]  $\leftarrow$  temp
23:    return false
24:  end procedure
25:  for row  $\leftarrow$  0 to rows do
26:    for col  $\leftarrow$  0 to cols do
27:      if dfs(row, col, 0) then
28:        return true
29:      end if
30:    end for
31:  end for
32:  return false
33: end procedure
```

---

## 2.2 Code

### 2.2.1 JavaScript

```
const exist = (board, word) => {
  if (!board || !word) return false;
  const rows = board.length;
  const cols = board[0].length;

  const dfs = (row, col, index) => {
    if (
      row < 0 || row >= rows ||
      col < 0 || col >= cols ||
      board[row][col] !== word[index]
    ) {
      return false;
    }
    if (index === word.length - 1) return true;

    const temp = board[row][col];
    board[row][col] = "/";
    const directions = [[-1, 0], [1, 0], [0, -1], [0, 1]];

    for (const [dx, dy] of directions) {
      if (dfs(row + dx, col + dy, index + 1)) return true;
    }

    board[row][col] = temp;
    return false;
  }

  for (let row = 0; row < rows; row++) {
    for (let col = 0; col < cols; col++) {
      if (dfs(row, col, 0)) return true;
    }
  }
  return false;
}
```