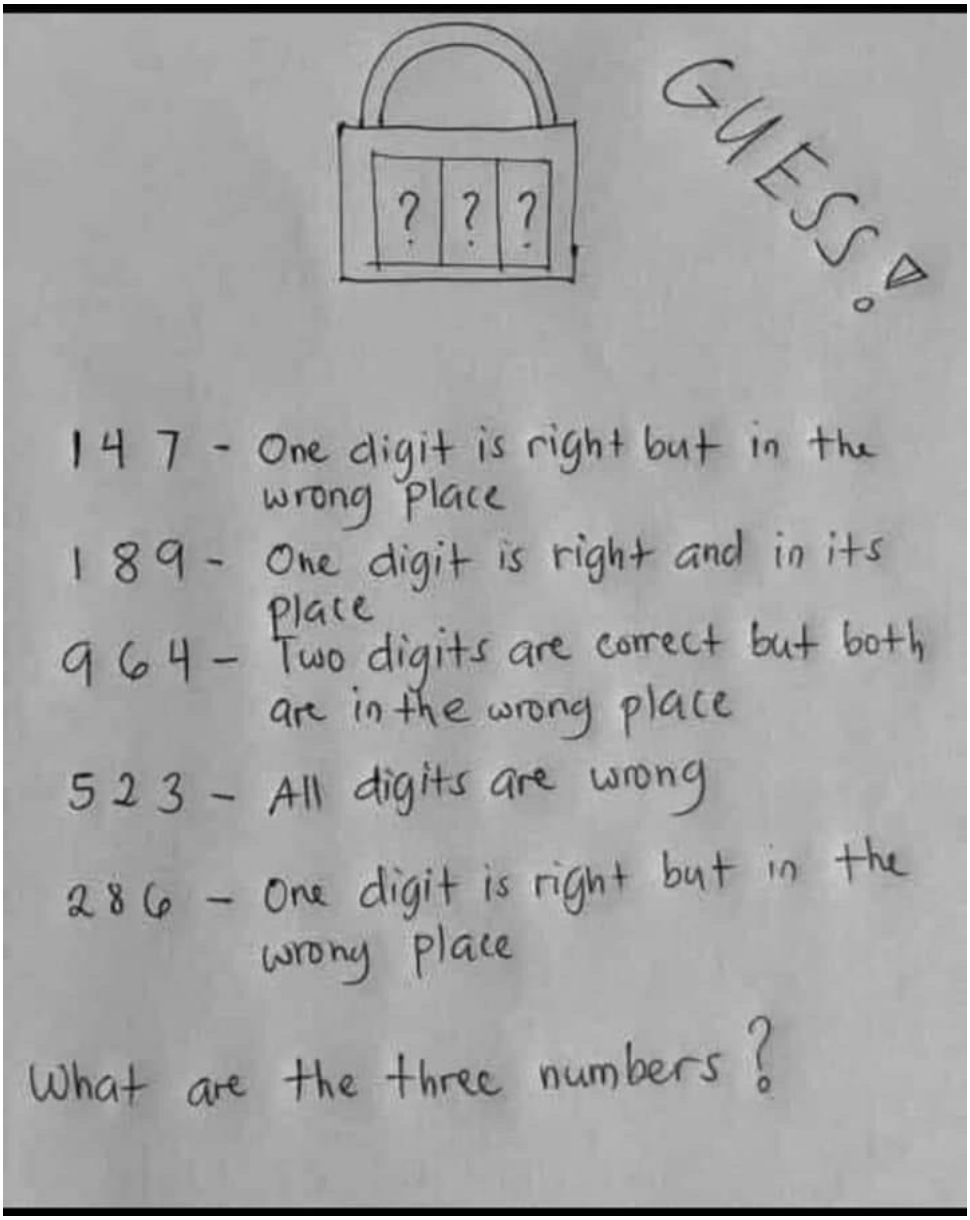
Puzzle 3 - Padlock

Please attempt this challenge and submit even if you only have a partial answer, we are keen to see your approach.

Write a code in a programming language of your choice to solve the following puzzle.



Ans. For this problem, I used JavaScript as a programming language.

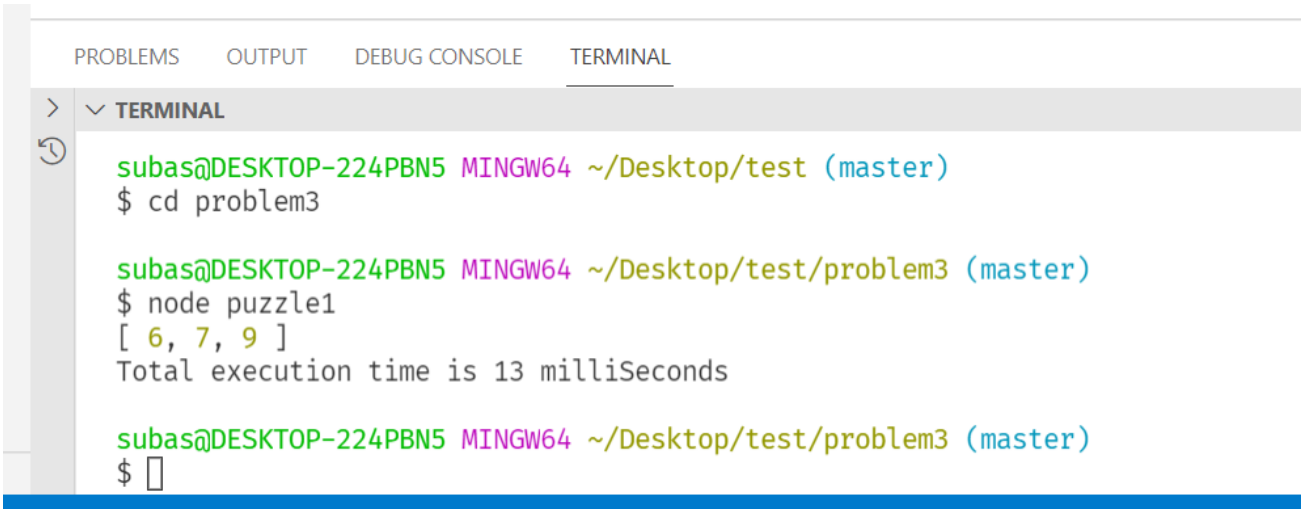
Source Code

```
let start = Date.now();

function totalCombinations(start, end){
    const allCombinations = [];
    for (let i = start; i <= end; i++) {
        for (let j = start; j <= end; j++) {
            for (let k = start; k <= end; k++) {
                if ((i !== j) && (i !== k) && (j !== k)) {
                    allCombinations.push([i, j, k])
                }
            }
        }
    }
    return allCombinations;
}

function OneDigitRightButWrongPlace(checkValue, allCombinations, numberO
fTestDigits) {
    return allCombinations.filter(combination => {
        const numberOfMatches = [];
        const numberOfEqualPositions = [];
        // Find matching items
        checkValue.forEach(value => {
            if (combination.includes(value)) {
                numberOfMatches.push(value)
            }
        });
        // Find matching position
        for (let i = 0; i < checkValue.length; i++) {
            if (combination[i] === checkValue[i]) {
                numberOfEqualPositions.push(checkValue[i]);
            }
        }
        // Check if matching items is number of test digits
        const numberOfOneMatches = numberOfMatches.length === numberOfTe
stDigits;

        // Check if number of matching position is 0
        const oneDifferentPosition = numberOfEqualPositions.length === 0
;
        return (numberOfOneMatches && oneDifferentPosition);
```

```
    })
}

function OneDigitRightAndRightPlace(checkValue, allCombinations, numberO
fTestDigits) {
    return allCombinations.filter(combination => {
        const numberOfMatches = [];
        const numberOfEqualPositions = [];
        // Find matching items
        checkValue.forEach(value => {
            if (combination.includes(value)) {
                numberOfMatches.push(value)
            }
        });
        // Find matching position
        for (let i = 0; i < checkValue.length; i++) {
            if (combination[i] === checkValue[i]) {
                numberOfEqualPositions.push(checkValue[i]);
            }
        }
        // Check if matching items is number of test digits
        const numberOfOneMatches = numberOfMatches.length === numberOfTe
stDigits;

        // Check if number of matching position is numberOfTestDigits
        const oneEqualPosition = numberOfEqualPositions.length === numbe
rOfTestDigits;
        return (numberOfOneMatches && oneEqualPosition);
    })
}
function TwoDigitsCorrectButWrongPlace(checkValue, allCombinations, numb
erOfTestDigits) {
    return allCombinations.filter(combination => {
        const numberOfMatches = [];
        const numberOfEqualPositions = [];
        // Find matching items
        checkValue.forEach(value => {
            if (combination.includes(value)) {
                numberOfMatches.push(value)
            }
        });
        // Find matching position
        for (let i = 0; i < checkValue.length; i++) {
            if (combination[i] === checkValue[i]) {
                numberOfEqualPositions.push(checkValue[i]);
```

```
            }
        }
        // Check if matching items is number of test digits
        const numberOfOneMatches = numberOfMatches.length === numberOfTe
stDigits;

        // Check if number of matching position is 0
        const oneEqualPosition = numberOfEqualPositions.length === 0;
        return (numberOfOneMatches && oneEqualPosition);
    })
}

function AllDigitAreWrong(checkValue, allCombinations) {
    return allCombinations.filter(combination => {
        let doesMatch = false;
        for (i = 0; i < checkValue.length; i++) {
            if (combination.includes(checkValue[i])) {
                doesMatch = true;
            }
        }
        return !doesMatch;
    })
}

const allCombinations = totalCombinations(1, 9);

const filter_1 = AllDigitAreWrong([5, 2, 3], allCombinations);
const filter_2 = OneDigitRightButWrongPlace([1, 4, 7], filter_1, 1);
const filter_3 = OneDigitRightAndRightPlace([1, 8, 9], filter_2, 1);
const filter_4 = TwoDigitsCorrectButWrongPlace([9, 6, 4], filter_3, 2);
const filter_5 = OneDigitRightButWrongPlace([2, 8, 6], filter_4, 1);

const result = filter_5[0];
console.log(result)

let end = Date.now();
const totalExecTime = end - start;
console.log(`Total execution time is ${totalExecTime} milliSeconds`);
```

Output:



Explanation:

1. totalCombinations(start, end) function is called which returns array of all possible 3-digits combination(Array) within range(start, end) with unique digits(i.e. 3 digits cannot be equal).
2. AllDigitAreWrong function is called where test check value(Array) and array of total combinations(obtained from step 1) is passed as argument and returns filtered array of combinations which doesn't include any of check value elements.
3. OneDigitRightButWrongPlace function is called which takes check value(Array) and array of filtered combinations(obtained from step 2) and returns combinations which satisfies one digit right and wrong place by comparing combinations with check value.
4. Similary, `OneDigitRightAndRightPlace`, `TwoDigitsCorrectButWrongPlace`, `OneDigitRightButWrongPlace` functions are called which finally returns single array with all matching conditions.
5. Finally, last filtered combination is stored in result variable and printed as well as execution time.