```
const fs = require('fs');
// Exercise 1 - Smallest Fraction Terms
function exercise1(num, den) {
    function gcd(num, den) {
        if (den === 0) {
            return num;
        } else {
            return gcd(den, num % den);
        }
    }

    let gcdValue = gcd(num, den);

    let reducedNum = num / gcdValue;
    let reducedDen = den / gcdValue;

    return [reducedNum, reducedDen];
}

// Exercise 2 - Magical Dates
function exercise2(day, month, year) {
    let product = day * month;

    let lastTwoDigits = year % 100;

return product === lastTwoDigits;
}


// Exercise 3 - All Sublists
function exercise3(l) {
    const result = [];

    function generateSublists(start, currentList) {
        if (start === l.length) {
            result.push(currentList.slice());
            return;
        }
        generateSublists(start + 1, currentList);

        if (Array.isArray(l[start])) {
            currentList.push(...l[start]);
        } else {
            currentList.push(l[start]);
        }
        generateSublists(start + 1, currentList);
        currentList.pop();
    }

    generateSublists(0, []);

    return result;
}


// Exercise 4 - English to Pig Latin Translator
function exercise4(word) {
    function Vowel(char) {
        return 'aeiou'.includes(char.toLowerCase());
    }

    if (Vowel(word[0])) {
        return word + "way";
    }

    for (let i = 0; i < word.length; i++) {
        if (Vowel(word[i])) {
            return word.slice(i) + word.slice(0, i) + "ay";
        }
    }

    return word;
}

//Exercise 5 - Morse Code Encoder
function exercise5(message) {
    const morseCodeMapping = {
      'A': '.-',    'B': '-...',  'C': '-.-.',  'D': '-..',   'E': '.',
      'F': '..-.',  'G': '--.',   'H': '....',  'I': '..',    'J': '.---',
      'K': '-.-',   'L': '.-..',  'M': '--',    'N': '-.',    'O': '---',
      'P': '.--.',  'Q': '--.-',  'R': '.-.',   'S': '...',   'T': '-',
      'U': '..-',   'V': '...-',  'W': '.--',   'X': '-..-',  'Y': '-.--',
      'Z': '--..',

      '0': '-----', '1': '.----', '2': '..---', '3': '...--', '4': '....-',
      '5': '.....', '6': '-....', '7': '--...', '8': '---..', '9': '----.'
```

```
    };

    message = message.toUpperCase();

    return message
      .split('')
      .map(char => morseCodeMapping[char] || '')
      .filter(morseChar => morseChar)
      .join(' ');
  }


// Exercise 6 - Spelling Out Numbers
function exercise6(num) {
    if (num < 0 || num > 999) {
        return "Number out of range";
    }

    const units = ["zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"];
    const teens = ["ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"];
    const tens = ["", "", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"];

    if (num === 0) {
        return units["zero"];
    }

    let words = "";

    if (Math.floor(num / 100) > 0) {
        if (num >= 100 && num < 200) {
            words += "a hundred";
        } else {
            words += units[Math.floor(num / 100)] + " hundred";
        }
        if (num % 100 > 0) {
            words += " and ";
        }
    }

    num %= 100;

    if (num > 19) {
        words += tens[Math.floor(num / 10)];
        if (num % 10 > 0) {
            words += "-" + units[num % 10];
        }
    } else if (num >= 10) {
        words += teens[num - 10];
    } else if (num > 0) {
        words += units[num];
    }

    return words;
}
// Exercise 7 - No Functions without Comments
function exercise7(filename) {
    const content = fs.readFileSync(filename, 'utf8');
    const lines = content.split('\n');
    const uncommentedfunction = [];

    for (let i = 1; i < lines.length; i++) {
        if (lines[i].trim().startsWith('function ')) {
            const previousLine = lines[i - 1].trim();
            if (!previousLine.startsWith('//')) {
                const functionName = lines[i].split(' ')[1].split('(')[0];
                uncommentedfunction.push(functionName);
            }
        }
    }

    return uncommentedfunction;
}

// Exercise 8 - Justify any Text
function exercise8(filename, length) {
    const text = fs.readFileSync(filename, 'utf8');
    const words = text.split(/\s+/);
    const lines = [];
    let currentLine = '';

    words.forEach(word => {
        if (currentLine.length + word.length + 1 > length) {
            lines.push(justifyLine(currentLine.trim(), length));
            currentLine = '';
        }
        currentLine += word + ' ';
    });
```

```
    if (currentLine.trim().length > 0) {
        lines.push(currentLine.trim());
    }

    return lines;
}

function justifyLine(line, length) {
    if (line.length >= length) return line;

    let gaps = line.split(/\s+/).length - 1;
    if (gaps === 0) return line;

    let spacesNeeded = length - line.replace(/\s+/g, '').length;
    let spaces = Array(gaps).fill(Math.floor(spacesNeeded / gaps));
    spacesNeeded -= spaces.reduce((a, b) => a + b, 0);

    for (let i = 0; spacesNeeded > 0; i = (i + 1) % gaps, spacesNeeded--) {
        spaces[i]++;
    }

    const words = line.split(/\s+/);
    let justifiedText = words[0];

    for (let i = 0; i < spaces.length; i++) {
        justifiedText += ' '.repeat(spaces[i]) + words[i + 1];
    }

    return justifiedText;
}

function readFileSync(filename, encoding) {
    try {
        return fs.readFileSync(filename, encoding);
    } catch (error) {
        console.error('Error reading file:', error);
        return '';
    }
}


function exercise9(start, end, moves) {

    return none;
}


// Exercise 9 - Knight's Challenge

function exercise9(start, end, moves) {

    return none;
}


// Exercise 10 - War of Species
function exercise10(environment) {
    return undefined;
}

module.exports = {
    // Exercise 1
    exercise1: exercise1,

    // Exercise 2
    exercise2: exercise2,

    // Exercise 3
    exercise3: exercise3,

    // Exercise 4
    exercise4: exercise4,

    // Exercise 5
    exercise5: exercise5,

    // Exercise 6
    exercise6: exercise6,

    // Exercise 7
    exercise7: exercise7,

    // Exercise 8
    exercise8: exercise8,

    // Exercise 9
```

```
    exercise9: exercise9,

    // Exercise 10
    exercise10: exercise10,
}
```