ADVENT OF CODE

DAY 1

```
var fs = require('fs');

var result = fs.readFileSync('adventOfCode/day1.txt','utf8')

var arr = result.split('\r\n')
var result = 0
var hashMap = {}
while (true){
   for (let num of arr) {
      result = result + parseInt(num)
      if (hashMap[result] === undefined) {
         hashMap[result] = true
      } else {
         console.log("GOT " + result)
         return true
      }
   }
}
```

DAY 2

```
/*  First Part
var fs = require('fs');

var result = fs.readFileSync('adventOfCode/day2.txt','utf8')

var arr = result.split('\r\n')
var twoSum = 0
var threeSum = 0

for (let id of arr) {
   var hashMap = {}
   for (let char of id) {
     if (hashMap[char] === undefined){
        hashMap[char] = 1
     } else {
        hashMap[char]++;
     }
   }
   if (Object.values(hashMap).includes(2)){
      twoSum++
   }
   if (Object.values(hashMap).includes(3)){
      threeSum++
   }
}

console.log(twoSum*threeSum)
*/

var fs = require('fs');

var result = fs.readFileSync('adventOfCode/day2.txt','utf8')

var arr = result.split('\r\n')

for (let i = 0; i < arr.length -1; i++){
   for (let j = i+1; j < arr.length; j++) {
      if (checkDifference(arr[i], arr[j]) === 1){
         console.log(commonLetters(arr[i], arr[j]));
      }
   }
}

function commonLetters(firstId, secondId) {
   let commonLetter = '';
   for (let i = 0; i < firstId.length; i++){
      if (firstId[i] === secondId[i]){
         commonLetter+= firstId[i]
      }
   }
   return commonLetter
}

function checkDifference(firstId, secondId) {
   let totalDiff = 0;
   for (let i = 0; i < firstId.length; i++){
      if (firstId[i] !== secondId[i]){
         totalDiff++
      }
   }
   return totalDiff
}
```

DAY 3

```
var fs = require('fs');

var result = fs.readFileSync('adventOfCode/day3.txt','utf8')

var claims = result.split('\r\n')
let fabric = {}
// Part 1
for (let claim of claims){
   let [claimId, claimSymbol, coordinates, dimensions] = claim.split(" ")
   let [xcoordinate, ycoordinate] = coordinates.slice(0, coordinates.length-1).split(",")
   let [width, height] = dimensions.split("x");
   xcoordinate = parseInt(xcoordinate)
   ycoordinate = parseInt(ycoordinate)
   width = parseInt(width)
   height = parseInt(height)

   for (let i = xcoordinate; i < xcoordinate + width; i++){
      for (let j = ycoordinate; j < ycoordinate + height; j++){
         if (fabric[`${i},${j}`]){
            fabric[`${i},${j}`]++
         } else {
            fabric[`${i},${j}`] = 1
         }
      }
   }
}
let overlap = 0
for (let index in fabric){
   if (fabric[index]>1){
      overlap++
   }
}



// Part2
for (let claim of claims){
   let [claimId, claimSymbol, coordinates, dimensions] = claim.split(" ")
   let [xcoordinate, ycoordinate] = coordinates.slice(0, coordinates.length-1).split(",")
   let [width, height] = dimensions.split("x");
   xcoordinate = parseInt(xcoordinate)
   ycoordinate = parseInt(ycoordinate)
   width = parseInt(width)
   height = parseInt(height)

   let nonOverlap = true
   for (let i = xcoordinate; i < xcoordinate + width; i++){
      for (let j = ycoordinate; j < ycoordinate + height; j++){
            if (fabric[`${i},${j}`] > 1){
                nonOverlap = false
            }
        }
    }

    if (nonOverlap){
        console.log(claimId)
    }
}
```

Day 4

```
var fs = require('fs');

var result = fs.readFileSync('adventOfCode/day4.txt','utf8')

var sleepMinutes = {}
var logs = result.split('\r\n')

// Part 1
logs = logs.sort()
let guardId = 0;
let startSleepMinute = 0
let endSleepMinute = 0

for (let log of logs){
    let [year ,timestamp, comment, id] = log.split(" ")
    if (comment.toLowerCase() === "guard") {
        guardId = parseInt(id.slice(1))

        if (sleepMinutes[guardId] == undefined) {
            sleepMinutes[guardId] = {}
            for (let i = 0; i < 60; i++){
                sleepMinutes[guardId][i] = 0
            }
        }
    }

    if (comment.toLowerCase() === "falls") {
        startSleepMinute = parseInt(timestamp.split(":")[1].slice(0,2));
    }

    if (comment.toLowerCase() === "wakes") {
        endSleepMinute = parseInt(timestamp.split(":")[1].slice(0,2));
        for (let i = startSleepMinute; i < endSleepMinute; i++){
            sleepMinutes[guardId][i]++
        }
    }
}

const mostAsleepGuard = Object.keys(sleepMinutes).map(guardSleepKey => {
    const guardSleep = sleepMinutes[guardSleepKey];
    const totalSleep = Object.values(guardSleep).reduce((prev, curr) => prev + curr, 0);

    return {
        guardId: guardSleepKey,
        totalSleep: totalSleep
    };
}).sort((a, b) => b.totalSleep - a.totalSleep)[0].guardId;

let mostAsleepMinute;
let currentMostAsleepMinute = 0;

Object.keys((sleepMinutes[mostAsleepGuard])).map(minute => {
    if (sleepMinutes[mostAsleepGuard][minute] > currentMostAsleepMinute) {
        currentMostAsleepMinute = sleepMinutes[mostAsleepGuard][minute];
        mostAsleepMinute = minute;
    }
});

console.log(mostAsleepGuard * mostAsleepMinute)


// Part2

let mostFrequentSleepMinute
let guardSleepID
let mostSleepCount = 0

Object.keys(sleepMinutes).map(guard => {
   let guardSleep = sleepMinutes[guard]
   Object.keys(guardSleep).map(minute => {
        if (mostSleepCount < guardSleep[minute]){
            guardSleepID = guard;
            mostFrequentSleepMinute = minute;
            mostSleepCount = guardSleep[minute];
        }
   });
});

console.log(mostFrequentSleepMinute * guardSleepID)
```

DAY 5

```
var fs = require('fs');

var result = fs.readFileSync('adventOfCode/day5.txt','utf8')

// var stack = []
// for (let char of result) {
//     if (!stack.length){
//         stack.push(char)
//         continue;
//     }
//     if ((char !== stack[stack.length-1]) && (char.toLowerCase() === stack[stack.length -
1].toLowerCase())) {
//         stack.pop()
//         continue
//     }
//     stack.push(char)
// }

let charMap = {
   1: ['a', 'A'],
   2: ['b', 'B'],
   3: ['c', 'C'],
   4: ['d', 'D'],
   5: ['e', 'E'],
   6: ['f', 'F'],
   7: ['g', 'G'],
   8: ['h', 'H'],
   9: ['i', 'I'],
   10: ['j', 'J'],
   11: ['k', 'K'],
   12: ['l', 'L'],
   13: ['m', 'M'],
   14: ['n', 'N'],
   15: ['o', 'O'],
   16: ['p', 'P'],
   17: ['q', 'Q'],
   18: ['r', 'R'],
   19: ['s', 'S'],
   20: ['t', 'T'],
   21: ['u', 'U'],
   22: ['v', 'V'],
   23: ['w', 'W'],
   24: ['x', 'X'],
   25: ['y', 'Y'],
   26: ['z', 'Z'],
}
```

```
let shortestLength;
for (let key in charMap){
    let newStringResult = result;
    for (let char of charMap[key]){
        // ONE WAY TO REPLACE USING regex
        var re = new RegExp(char,"g");
        newStringResult = newStringResult.replace(re, '')
        // newStringResult = newStringResult.split(char).join('')
    }
    var stack = []
    for (let char of newStringResult) {
        if (!stack.length){
            stack.push(char)
            continue;
        }
        if ((char !== stack[stack.length-1]) && (char.toLowerCase() === stack[stack.length -
1].toLowerCase())) {
            stack.pop()
            continue
        }
        stack.push(char)
    }
    if (!shortestLength){
        shortestLength = stack.length
    } else {
        if (shortestLength > stack.length){
            shortestLength = stack.length
        }
    }
}

console.log(shortestLength)
```

DAY 7

```
var fs = require('fs');

var result = fs.readFileSync('adventOfCode/day7.txt','utf8').split("\r\n")

nodes = {}
for (let line of result) {
    let dependentOn = line.split(" ")[1]
    let step = line.split(" ")[7]
    if (!nodes[step]){
        nodes[step] = [dependentOn]
    } else {
        nodes[step].push(dependentOn)
    }
    if (!nodes[dependentOn]){
        nodes[dependentOn] = []
    }
}

let stepsOrder = ''
let deletekey;
let nodesKeyLength = Object.keys(nodes).length
let sortedNodes = Object.keys(nodes).sort()
while (stepsOrder.length < nodesKeyLength){
    for (let i = 0; i < sortedNodes.length; i++){
        let key = sortedNodes[i]
        if (nodes[key].every(node => stepsOrder.includes(node))){
            stepsOrder = stepsOrder + key
            deletekey = i
            break
        }
    }
    sortedNodes.splice(deletekey, 1)
}
console.log(stepsOrder)
```

DAY 8

```
var fs = require('fs');

var result = fs.readFileSync('adventOfCode/day8.txt','utf8').split(" ")


// Part 1
function convertToTree(data){
    let childrenCount = parseInt(data[0])
    let metadataCount = parseInt(data[1])

    if (childrenCount === 0){
        let metadata = data.slice(2, 2 + metadataCount);
        let length = 2 + metadataCount;
        return {
            children: [],
            metadata,
            length
        }
    }
    let children = []
    let pointer = 2;
    while (childrenCount > 0) {
        let child = convertToTree(data.slice(pointer))
        children.push(child);
        pointer = pointer + child.length;
        childrenCount--
    }

    let childrenLength = children.reduce((acc, v) => {
        return (acc + v.length)
    }, 0)
    let metadata = data.slice(2 + childrenLength, 2 + childrenLength + metadataCount);
    return {
        children,
        metadata,
        length: childrenLength + 2 + metadataCount
    }
}

const rootNode = convertToTree(result);
const sumMetadata = (node) => {
  const localSum = node.metadata.reduce((acc, v) => {
      return (acc + parseInt(v))
    }, 0);
  if (node.children.length === 0) {
    return localSum;
  }
  return localSum + node.children.reduce((acc, v) => acc + sumMetadata(v), 0);
}

// console.log(sumMetadata(rootNode));
// Part 2
const rootNodeSum = (node) => {
    if (node.children.length === 0){
        return node.metadata.reduce((acc, v) => acc + parseInt(v), 0);
    }

    return node.metadata.reduce((acc, v) => {
        if (!v || v > node.children.length){
            return acc;
        } else {
            return acc + rootNodeSum(node.children[v - 1])
        }
    }, 0);
}

console.log(rootNodeSum(rootNode))
```

DAY 9

```
var fs = require('fs');

var result = fs.readFileSync('adventOfCode/day9.txt','utf8').split(" ")

// Part 1
let currentNode = { val:0, next:null, prev:null }
currentNode.next = currentNode;
currentNode.prev = currentNode;

let numberOfPlayers = 430
let marbleCount = 7158800
let playerScore = [];
for (let i = 0; i < numberOfPlayers; i++){
  playerScore[i] = 0
}

for (let marbleValue = 1; marbleValue <= marbleCount; marbleValue++){
  let i = (marbleValue - 1)%numberOfPlayers
  if (marbleValue%23 === 0){
    let removeNode = currentNode.prev.prev.prev.prev.prev.prev.prev
    playerScore[i] = playerScore[i] + marbleValue + removeNode.val
    removeNode.prev.next = removeNode.next
    removeNode.next.prev = removeNode.prev
    currentNode = removeNode.next
  } else {
    let newNode = {val : marbleValue}
    let marble1 = currentNode.next
    let marble2 = currentNode.next.next
    marble1.next = newNode
    newNode.next = marble2
    marble2.prev = newNode
    newNode.prev = marble1
    currentNode = newNode
  }
}

console.log(Math.max(...playerScore))
```