Programming Fundamentals

Lab Manual - Week 08

Introduction

Welcome to your favorite programming Lab. In this lab manual, we will revise all the previous programming concepts and solve the problems.

Let's do some coding.

Task 01: Write the function that takes three dimensions of a brick: height(a), width(b) and depth(c) and returns true if this brick can fit into a hole with the width(w) and height(h).

Test Cases:

doesBrickFit(1, 1, 1, 1, 1) ➡ true

doesBrickFit(1, 2, 1, 1, 1) ➡ true

doesBrickFit(1, 2, 2, 1, 1) ➡ false

Notes:

- You can turn the brick with any side towards the hole.
- We assume that the brick fits if its sizes equal the ones of the hole (i.e. brick size should be equal to the size of the hole).
- You can't put a brick in at a non-orthogonal angle.

Task 02: It's a Pokemon battle! Your task is to calculate the damage that a particular move would do using the following formula (not the actual one from the game):

damage = 50 * (attack / defense) * effectiveness

- attack = your attack power
- defense = the opponent's defense
- effectiveness = the effectiveness of the attack based on the matchup (see explanation below)

Effectiveness:

Attacks can be super effective, neutral, or not very effective depending on the matchup. For example, water would be super effective against fire, but not very effective against grass.

- Super effective: 2x damage
- Neutral: 1x damage
- Not very effective: 0.5x damage

To prevent this challenge from being tedious, you'll only be dealing with four types: fire, water, grass, and electric. Here is the effectiveness of each matchup:

- fire > grass
- fire < water
- fire = electric
- water < grass
- water < electric
- grass = electric

The function you must implement takes in:

- your type
- the opponent's type
- your attack power
- the opponent's defense

Test Cases:

calculateDamage("fire", "water", 100, 100) ➞ 25

calculateDamage("grass", "fire", 35, 5) ➞ 175

calculateDamage("electric", "fire", 100, 100) ➞ 50

Notes

- Any type against itself is not very effective. Also, assume that the relationships between different types are symmetric (if A is super effective against B, then B is not very effective against A).

Task 03: You have cultivated a plant, and after three long months, the time has come to reap the fruits (or the flowers, in this case) of your hard work. During the growth phase, you added water and fertilizer, and kept a constant temperature. It's time to check how much the plant has grown!

Programming Fundamentals

Lab Manual - Week 08

A plant is represented horizontally, from the base to the left, to the end to the right:

---@---@---@

The stem is made of hyphens, and the flowers are represented by symbols. A plant always starts with the stem, and always ends with flowers.

The four given parameters are:

- **seed (string)** determines the type of flowers generated by the plant.
- **water (integer)** each unit of water extends the portion of stem between the flowers, and gives the total number of segments (stem + flowers) of the plant.
- **fert (integer)** each unit of fertilizer increases the amount of flowers, grouped in clusters.
- **temp (integer)** if the temperature recorded is between 20°C and 30°C (bounds included) the plant grows normally, otherwise all the flowers die, except for a single survivor at the end of the stem.

Given the above parameters, implement a function that returns a string representing the plant.

Test Cases:

plant("@", 3, 3, 25) ➡ "---@@@---@@@---@@@"

// Water gives the length of the stem portions between flowers.
// Water gives the total number of segments.
// Fertilizer gives the length of flowers clusters.
// In this case the temperature is in the acceptable range 20°C | 30°C

plant("#", 1, 5, 30) ➡ "-#####"

plant("&", 5, 1, 20) ➡ "-----&-----&-----&-----&-----&"

plant("§", 3, 3, 15) ➡ "---------§"

// The temperature out of range make all flowers die, except the last one.
// The stem is not affected by temperature.

Notes:

- All given cases will have valid parameters for water and fert, you have to only check that temp is in the "safe" range (20°C - 30°C).

Task 04: Create a function that finds how many prime numbers there are, up to the given integer.

Examples

primeNumbers(10) ➡ 4

// 2, 3, 5 and 7

primeNumbers(20) ➡ 8

// 2, 3, 5, 7, 11, 13, 17 and 19

primeNumbers(30) ➡ 10

// 2, 3, 5, 7, 11, 13, 17, 19, 23 and 29

Task 05: A positive number's population is the sum of 1's in its binary representation.

- An **evil** number has an even numbered population.
- An **odious** number has an odd numbered population.
- A number is **pernicious** if its *population* is a prime number.

Create a function that takes a number as an argument and returns a string of all its descriptors ("Evil", "Odious", or "Pernicious").

Examples

howBad(7) ➡ "Odious, Pernicious"

// 7 in binary is "111".
// 1 + 1 + 1 = 3 in "111" means "Odious" should be added to the array answer.
// 3 is a prime number, meaning "Pernicious" should also be added.

howBad(17) ➞ "Evil, Pernicious"

// 17 in binary is "10001".
// 1 + 1 = 2 in "10001" means "Evil" should be added to the array answer.
// 2 is a prime number, meaning "Pernicious" should also be added.

howBad(23) ➞ "Evil"

// 23 in binary is "10111".
// Four 1's in "10111" means "Evil" should be added to the array answer.
// 4 is not a prime number, meaning "Pernicious" should not be added.

Task 06: **Mubashir** needs your help to construct a building which will be a pile of n cubes. The cube at the bottom will have a volume of **n^3**, the cube above will have volume of **(n-1)^3** and so on until the top which will have a volume of **1^3**.

Given the total volume mof the building, can you find the number of cubes n required for the building?
In other words, you have to return an integer n such that:
n^3 + (n-1)^3 + ... + 1^3 == m
Return - 1 if there is no such number.

Test Cases:

pileCubes(1071225) ➞ 45

pileCubes(4183059834009) ➞ 2022

pileCubes(16) ➞ -1

Good Luck and Best Wishes !!

Happy Coding ahead :)