WEEK 3 - ASSIGNMENT 1 REVISION

NOTE:

- **No need to submit anywhere, just keep track of all the codes you have written in a specific folder.**
- **Compare your solution with the solution I'll provide, in case of doubts, kindly reach out to me.**
- **You may get assignment solution in format of PDF or VIDEO solution, depending on the difficulty level.**

Q1. Write a function that converts hours into seconds.

Examples:

- how_many_seconds(2) ➞ 7200
- how_many_seconds(10) ➞ 36000
- how_many_seconds(24) ➞ 86400

60 seconds in a minute, 60 minutes in an hour

Don't forget to return your answer.

Q2. Create a function that takes the age in years and returns the age in days.

Examples:

- calc_age(65) ➞ 23725
- calc_age(0) ➞ 0
- calc_age(20) ➞ 7300

Use 365 days as the length of a year for this challenge.

Ignore leap years and days between last birthday and now.

Expect only **positive** integer inputs.

Q3. Create a function that takes a base number and an exponent number and returns the calculation.

Examples:

- calculate_exponent(5, 5) ➞ 3125
- calculate_exponent(10, 10) ➞ 10000000000
- calculate_exponent(3, 3) ➞ 27

All test inputs will be positive integers.

Don't forget to return the result.

Q4. Create a function that takes the number of **wins**, **draws** and **losses** and calculates the number of points a football team has obtained so far.

- wins get 3 points
- draws get 1 point
- losses get 0 points

Examples:

- football_points(3, 4, 2) ➞ 13
- football_points(5, 0, 2) ➞ 15
- football_points(0, 0, 1) ➞ 0

Inputs will be numbers greater than or equal to 0.

Q5. A farmer is asking you to tell him how many legs can be counted among all his animals. The farmer breeds three species:

- **chickens** = 2 legs
- **cows** = 4 legs
- **pigs** = 4 legs

The farmer has counted his animals and he gives you a subtotal for each species. You have to implement a function that returns the total number of legs of all the animals.

Examples:

- animals(2, 3, 5) ➞ 36

- animals(1, 2, 3) ➞ 22
- animals(5, 2, 8) ➞ 50

Don't forget to **return** the result.

The order of animals passed is **animals(chickens, cows, pigs).**

Q6. Write a function that takes two integers (**hours**, **minutes**), converts them to **seconds**, and adds them.

Examples:

- convert(1, 3) ➞ 3780
- convert(2, 0) ➞ 7200
- convert(0, 0) ➞ 0

Don't forget to return the result.

Q7. Write a function that returns the string **"something"** joined with a space " " and the given argument **a**.

Examples:

- give_me_something("is better than nothing") ➞ "something is better than nothing"
- give_me_something("Bob Jane") ➞ "something Bob Jane"
- give_me_something("something") ➞ "something something"

Q8. Create a function that takes two arguments: the original **price** and the **discount** percentage as integers and returns the final price after the discount.

Examples:

- dis(1500, 50) ➞ 750
- dis(89, 20) ➞ 71.2
- dis(100, 75) ➞ 25

Your answer should be rounded to two decimal places.

Q9. In this challenge, establish if a given integer **num** is a Curzon number. If **1** plus **2** elevated to **num** is exactly divisible by **1** plus **2** multiplied by **num**, then **num** is a Curzon number.

Given a non-negative integer **num**, implement a function that returns **True** if **num** is a Curzon number, or **False** otherwise.

Examples:

- is_curzon(5) ➞ True
    - # 2 ** 5 + 1 = 33
    - # 2 * 5 + 1 = 11
    - # 33 is a multiple of 11

- is_curzon(10) ➞ False
    - # 2 ** 10 + 1 = 1025
    - # 2 * 10 + 1 = 21
    - # 1025 is not a multiple of 21

- is_curzon(14) ➞ True
    - # 2 ** 14 + 1 = 16385
    - # 2 * 14 + 1 = 29
    - # 16385 is a multiple of 29

Q10. Create a function that takes the number of daily average recovered cases **recovers**, daily average **new_cases**, current **active_cases**, and returns the number of **days** it will take to reach zero cases.

Examples:

- end_corona(4000, 2000, 77000) ➞ 39
- end_corona(3000, 2000, 50699) ➞ 51
- end_corona(30000, 25000, 390205) ➞ 79

The number of people who recover per day **recovers** will always be greater than daily **new_cases**.

Be conservative and round up the number of days needed.

Q11. Create a function that takes **damage** and **speed** (attacks per second) and returns the amount of damage after a given **time**.

Examples:

- damage(40, 5, "second") ➞ 200
- damage(100, 1, "minute") ➞ 6000
- damage(2, 100, "hour") ➞ 720000

Return **"invalid"** if damage or speed is negative.

Q12. Create a function that takes three arguments **a, b, c** and returns the sum of the numbers that are evenly divided by **c** from the range **a, b** inclusive.

Examples:

- evenly_divisible(1, 10, 20) ➞ 0
    - # No number between 1 and 10 can be evenly divided by 20.

- evenly_divisible(1, 10, 2) ➞ 30
    - # 2 + 4 + 6 + 8 + 10 = 30

- evenly_divisible(1, 10, 3) ➞ 18
    - # 3 + 6 + 9 = 18

Return 0 if there is no number between a and b that can be evenly divided by c.

Q13. Create a function that returns the **thickness (in meters)** of a piece of paper after folding it **n** number of times. The paper starts off with a thickness of 0.5mm.

Examples:

- num_layers(1) ➞ "0.001m"
    - # Paper folded once is 1mm (equal to 0.001m)

- num_layers(4) ➞ "0.008m"
    - # Paper folded 4 times is 8mm (equal to 0.008m)

- num_layers(21) ➞ "1048.576m"
    - # Paper folded 21 times is 1048576mm (equal to 1048.576m)

There are 1000mm in a single meter.

Don't round answers.

Q14. Create a function that takes two parameters and, if both parameters are strings, **add them** as if they were integers or if the two parameters are integers, **concatenate them.**

- stupid_addition(1, 2) ➞ "12"
- stupid_addition("1", "2") ➞ 3
- stupid_addition("1", 2) ➞ None

If the two parameters are different data types, return None.

All parameters will either be strings or integers.

Q15. Create a function that takes three values:

- **h** hours
- **m** minutes
- **s** seconds

Return the value that's the longest duration.

Examples:

- longest_time(1, 59, 3598) ➞ 1
- longest_time(2, 300, 15000) ➞ 300
- longest_time(15, 955, 59400) ➞ 59400

No two durations will be the same.

Q16. Create a function which takes two strings (**p1** and **p2** — which represent player 1 and 2) as arguments and returns a string stating the winner in a game of Rock, Paper, Scissors.

Each argument will contain a single string: **"Rock"**, **"Paper"**, or **"Scissors"**. Return the winner according to the following rules:

- **Rock** beats **Scissors**
- **Scissors** beats **Paper**
- **Paper** beats **Rock**

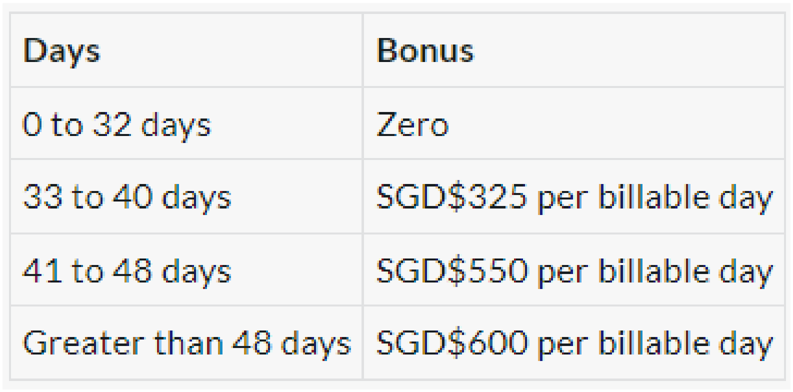
If **p1** wins, return the string **"The winner is p1"**. If **p2** wins, return the string **"The winner is p2"** and if **p1** and **p2** are the same, return **"It's a draw".**

Examples:

- rps("Rock", "Paper") ➞ "The winner is p2"
- rps("Scissors", "Paper") ➞ "The winner is p1"
- rps("Paper", "Paper") ➞ "It's a draw"

Q17. A financial institution provides professional services to banks and claims charges from the customers based on the number of man-days provided. Internally, it has set a scheme to motivate and reward staff to meet and exceed targeted billable utilization and revenues by paying a bonus for each day claimed from customers in excess of a threshold target.

This quarterly scheme is calculated with a threshold target of 32 days per quarter, and the incentive payment for each billable day in excess of such threshold target is shown as follows:



Days	Bonus
0 to 32 days	Zero
33 to 40 days	SGD$325 per billable day
41 to 48 days	SGD$550 per billable day
Greater than 48 days	SGD$600 per billable day

Please note that incentive payment is calculated progressively. As an example, if an employee reached total billable days of 45 in a quarter, his/her incentive payment is computed as follows:

32*0 + 8*325 + 5*550 = 5350

Write a function to read the billable days of an employee and return the bonus he/she has obtained in that quarter.

Examples:

- bonus(15) ➞ 0
- bonus(37) ➞ 1625

- bonus(50) ➞ 8200

Q18. Create a function that finds how many prime numbers there are, up to the given integer.

- prime_numbers(10) ➞ 4
  - # 2, 3, 5 and 7

- prime_numbers(20) ➞ 8
  - # 2, 3, 5, 7, 11, 13, 17 and 19

- prime_numbers(30) ➞ 10
  - # 2, 3, 5, 7, 11, 13, 17, 19, 23 and 29

Q19. Create a function that takes an integer **n** and returns the **factorial of factorials.** See below examples for a better understanding:

Examples:

- fact_of_fact(4) ➞ 288
  - # 4! * 3! * 2! * 1! = 288

- fact_of_fact(5) ➞ 34560

- fact_of_fact(6) ➞ 24883200

Q20. Given an integer, create a function that returns the next prime. If the number is prime, return the number itself.

Examples:

- next_prime(12) ➞ 13

- next_prime(24) ➞ 29

- next_prime(11) ➞ 11
  - # 11 is a prime, so we return the number itself.