

✓ AddisCoder Quiz 2

Congrats on completing Week 2! 🥳

You will have **2 hours** to complete this written quiz. You may use any method to solve a problem unless it explicitly asks for a recursive solution. The questions are not necessarily in order of difficulty.

Please write your **full name** and **lab number** (NB111, NB112, NB114, Green Lab, or Brown Lab) on the first page of the quiz.

This quiz will cover material from weeks 1 and 2, including:

- Conditionals and Loops
- Lists and Dictionaries
- Functions
- Recursion
- Images

✓ Problem 1: Report Card Day! (Dictionaries)

1.1 Create an empty dictionary named `report_card`.

1.2 To the `report_card` dictionary, add the following key-value pairs:

"Maths" -> 95

"Physics" -> 88

"Chemistry" -> 92

1.3 You aced the Math final! Change the value of "Maths" to 100 in the report_card dictionary, and print the updated dictionary.

1.4 Using the report_card dictionary, print your grade in Chemistry class.

✓ Problem 2: Mutation Mayhem (Lists and Dictionaries) 📖

2.1 What is the **printed output** of the following code?

```
A = [[2], 12, [[4], 6], 8, 10]
```

```
B = A
```

```
B.pop()
```

```
A[2][1] = 9
```

```
A[0] += [7]
```

```
print(A)
```

```
print(B)
```

Output:

2.2 What is the **printed output** of the following code?

```
sales = {"Bunna": 30, "Shai": 15, "Mrinda": 55}

sales["Mrinda"] = sales["Bunna"] + 20
removed = sales.pop("Shai")
sales["Milk"] = removed + sales["Mrinda"]
sales["Bunna"] -= 5

print(sales)
```

Output:

✓ Problem 3: Loopy Logic (Print with Loops)

3.1 What is the **printed output** of the following code?

```
for i in range(2 ** 2 - 4 // 2):
    x = i + 1
    if x <= 5:
        print(i + 1 * 2)
```

Output:

3.2 What is the **printed output** of the following code?

```
for i in range(2, 5, 2):
    line = str(i) + ": "

    for j in range(1, i+1):
        if i % j == 0:
            line += str(j) + " "
    print(line)
```

Output:

✓ Problem 4: Dictionary Building

4.1 Write a function `word_length(word)` that takes a single string `word` and returns the number of letters in the word.

Example:

```
word_length("injera") -> 6
```

```
word_length("teff") -> 4
```

Here, "injera" has 6 letters, so the function returns 6 . Similarly, "teff" has 4 letters, so the function returns 4 .

4.2 Write a function `group_by_length(words)` that takes a list of strings `words` and returns a dictionary where:

- The keys are the lengths of the words.
- The values are lists of words that have that length.

Example:

```
group_by_length(["injera", "teff", "kitfo", "shiro", "tibs"]) -> {6: ["injera"], 4: ["teff", "tibs"], 5: ["kitfo", "shiro"]}
```

Here, `injera` has 6 letters, so it is grouped under the key 6. Both `teff` and `tibs` have 4 letters, so they are grouped together under key 4. Similarly, `kitfo` and `shiro` each have 5 letters and are grouped under key 5.

✓ Problem 5: Find the Black Pixel 🕵️

5.1 What is the **printed output** of the following code?

The image below is a 5×5 grid where all pixels are white except for one black pixel (RGB = 0, 0, 0).



Write a function `find_black_pixel(img)` that takes an image of arbitrary size that contains exactly one black pixel and returns the (x, y) coordinates of the black pixel.

You can assume:

- The input image is square (N×N) and can be of any size.
- The image contains exactly one black pixel. (Not necessarily in the same spot as the example)
- All other pixels are white (RGB = 255, 255, 255).

Hint: You can use `img.width` and `img.height` to get width and height of the input image.

✓ Problem 6: Filters Galore 🌈

Consider the following code that inverts the colors of an image using the SimpleImage library.

```
from simpleimage import SimpleImage

def invert(img):
    for pixel in img:
        pixel.red = 255 - pixel.red
        pixel.green = 255 - pixel.green
        pixel.blue = 255 - pixel.blue
```

Suppose you run this code twice in a row on the same image:

```
image = SimpleImage("assets/einstein.jpeg")
invert(image)
invert(image)
```

What will the final image look like? Select one of option below.

- ☐ A. It will be all white
- ☐ B. It will be the negative (color-inverted) version of the original image
- ☐ C. It will be back to the original image
- ☐ D. It will be all black

✓ Problem 7: Product Puzzle

7.1 What is the **printed output** of the following code?

```
def puzzle_product(x):  
    result = 1  
    for i in x:  
        result *= i  
    return result  
  
def fun_func(x):  
    result = []  
    for i in x:  
        result += [puzzle_product(i)]  
    return result  
  
print(fun_func([[1,2,3], [2,3,4], [3,4,5]]))
```

Answer: _____

✓ Problem 8: Self-Dividing Numbers

A self-dividing number is an integer n that:

- contains no digit 0 and
- is divisible by every one of its digits.

Example:

- 128 is self-dividing → 128 is divisible by 1, 128 is divisible by 2, and 128 is also divisible by 8
- 26 is not self-dividing → digit 6 does not divide 26

8.1 First, write a function called `digits_of(n)` that returns the list of decimal digits of n (in any order).

Example:

`digits_of(128) → [1, 2, 8]`

`digits_of(26) → [2, 6]`

8.2 Using your `digits_of(n)` function from Problem 8.1, write a function called `is_self_dividing(n)` that returns `True` if `n` is a self-dividing number, and `False` otherwise.

Examples:

`is_self_dividing(128)` → `True` since digits: `[1, 2, 8]`; 128 is divisible by all

`is_self_dividing(26)` → `False` since digits: `[2, 6]`; 26 is not divisible by 6

`is_self_dividing(101)` → `False` since digits: `[1, 0, 1]`; contains a 0 → not self-dividing

8.3 Write a function `print_self_dividing(n)` that prints every self-dividing number from 1 to `n` (inclusive), one per line. You can use the functions you wrote in the previous subproblems.

✓ Problem 9: Debugging

The code below crashes with the error: `TypeError: unsupported operand type(s) for *: 'int' and 'NoneType'`

```
def factorial(n):  
    if n == 0:  
        return  
    return n * factorial(n - 1)  
  
factorial(5)
```

Circle the line of code with the bug in it. Then, rewrite that line of code to fix the bug.

Answer: _____

✓ Problem 10: Recursion

The following is a recursive function that takes a list of numbers as an argument and returns a single number.

```
def mystery_function(numbers):  
    if len(numbers) == 0:  
        return 0  
    else:  
        last_number_sq = numbers[-1] ** 2  
        numbers.pop() # remove last number  
        value = mystery_function(numbers)  
  
        if last_number_sq % 2 == 0:  
            return value + last_number_sq  
        else:  
            return value
```

10.1 Circle the **base case** of the function.

10.2 In your own words, explain what the function `mystery_function` does.

Answer:

10.3 What is the output of `mystery_function([3,4,5,2,1])`?

Output: _____

✓ Problem 11: Recursion Tracing 🔍

```
def shout(n):  
    print("Shout!")  
    if n > 1:  
        shout(n - 1)  
        shout(n - 2)
```

11.1 How many times is Shout! printed when shout(2) is called?

Answer: _____

11.2 How many times is Shout! printed when shout(3) is called?

Answer: _____

✓ Problem 12: Recursion Debugging 🖥️

```
def nested_list_sum(lst):  
    s = 0  
    for x in lst:  
        if type(x) == int:  
            s += x  
        else:  
            nested_list_sum(x)  
    return s
```

The function above is meant to compute the sum of all integers in a (possibly nested) list. However, `nested_list_sum([1, 2, 3, [4, 5]])` returns 6 instead of the correct result 15. **Circle the buggy line and rewrite it with the correct code.**

Answer: _____

✓ Problem 13: Recursively Remove All Digits

Write a recursive function `remove_digits(s)` that removes all numeric digits from the string `s`.

Example:

```
remove_digits("a1b2c3") → "abc"
```

```
remove_digits("123abc") → "abc"
```

```
remove_digits("no digits") → "no digits"
```

Hint: You can use the function `isdigit()` to check if a string consists entirely of digits. For example, `"10013".isdigit()` and `"4".isdigit()` return `True`. On the other hand, `"abc".isdigit()` and `"ab2c".isdigit()` return `False`, since they contain non-digit characters.

⚠ Restrictions: You must use recursion. You **cannot** use any loops (for or while).

✓ Problem 14: Sum Cubes of Even Numbers... Using Recursion! 🧊

Write a function `sum_cubes_even(n)` that recursively computes the sum of the cubes of all even numbers from 1 up to and including `n`, if `n` is even. If `n` is odd, include only even numbers less than `n`.

You are guaranteed that `n` is a positive integer.

Mathematically, the function should evaluate:

$2^3 + 4^3 + 6^3 + \dots + m^3 \rightarrow$ where `m` is the largest even number less than or equal to `n`.

⚠ Restrictions: You must use recursion. You **cannot** use any loops (for or while).

Example:

`sum_cubes_even(5)` = 72 because: $2^3 + 4^3 = 8 + 64 = 72 \rightarrow$ `m` is 4 in this case, which is the largest even number less than or equal to 5

`sum_cubes_even(6)` = 288 because: $2^3 + 4^3 + 6^3 = 8 + 64 + 216 = 288 \rightarrow$ `m` is 6, which is equal to `n`.

`sum_cubes_even(1)` = 0 (no even numbers up to 1)

✓ Bonus Problem ✨

This problem is not counted towards the quiz. Only answer it if you answered everything else.

How many times is "Hello" printed when the following code is run?

```
def fun(x):  
    if x == 0:  
        return  
    if x%2 == 0:  
        print("Hello")  
        fun(x-1)  
        fun(x-2)  
    else:  
        fun(x-1)  
        fun(x-1)
```

```
fun(10)
```

Answer: _____