המחלקה להנדסת חשמל ומחשבים
יסודות מדעי המחשב
סמסטר ב' התשפ"ב

Homework Assignment 3

General information:

Responsible teaching assistant: Hadar Cohen.

Students' Questions regarding the assignment will be answered only in the HW3 forum.

If necessary, office hours are on Thursday, 16:00-17:00. Please send an email to coordinate prior (chada@post.bgu.ac.il).

Pay Attention:

- **In this assignment,** you can assume the input is valid for every question.
- Do not use external libraries to solve this exercise (no import statements allowed).
- All functions should be implemented using **only recursion**; therefore, any loops are not allowed (i.e., no *for*, *while* statements).
- As mentioned, some tests will be visible for your convenience, and others will not.
- You must code the assignment by yourself. Similarity tests will be performed automatically, and similar codes will be automatically graded as 0.
- In some of the questions, you may need to define auxiliary functions which should be recursive, while the question function provided to you might not and will be used only as a wrapper function. For example:

```
def questionX(num: int) -> bool:
    '''
    Call your recursive function for QsX
    return its Boolean value
    '''
    # WRITE YOUR CODE HERE!
    questionX_rec(var1, var2, ...)
```

- Template code with the main block code is given. Do not modify the template!
- In the provided functions, write the code only under "# WRITE YOUR CODE HERE!" comment.
- Good luck!

Question 1 – Perfect Number

A Perfect Number is a positive integer that equals the sum of its positive divisors (not including itself).

For example: the number 6 divisors are 1, 2, 3, and since 1 + 2 + 3 = 6, we say that 6 is a perfect number.

Implement the function **question1(num)** that receives an integer number **num** as an input. The function returns the Boolean value *True* if the number is a Perfect number and *False* otherwise.

Note:

- Your implantation must be recursive.

Running examples:

```
print(question1(num=28))
```

since the divisors of 28 are 14, 7, 4, 2, 1, the returned value would be *True*.

```
print(question1(num=8))
```

since the divisors of 8 are 4, 3, 1, the returned value would be *False*.

Question 2 – Subset Sum Problem

Given a set of integers, S, in the subset sum problem, the objective is to decide whether there exists a sum of a subset of numbers $s \subseteq S$ (Where S can be a multiset, i.e., the same number can occur more than once) Such that their sum is equal to an integer X.

Implement the function **question2(lst, x)** that receives a list of integers, **lst**, representing a multiset, and an integer, **x**, representing the target sum. The function returns the number of subsets of **lst** such that their sum is **x**.

Note:

- The sum of an empty set (or subset) is 0.
- We will consider each element in the set as a unique element. Even if two elements are equal, they will be considered as unique elements.
- The return value is an integer.
- Your implantation must be recursive.

Running examples:

```
print(question2(lst=[7, 6, 1, 20], x=14))
```

since only one subset (7, 6, 1) has sum 14, the returned value is 1.

```
print(question2(lst=[7, 6, 1, 20, 1], x=14))
```

since two subsets (two {7, 6, 1} – the number 1 has two occurrences that are considered unique by our definition), the returned value is 2.

```
print(question2(lst=[7, 6, 1, 20, 1, 50, 100, 200], x=1000))
```

since there is no subset of sum 1000, the returned value is 0.

Question 3 – Epidemic Spread

A. You are asked to implement the function **question3_a(mat, indices, epidemic)** that simulates the spreading of an epidemic.
You are given as input to the function:

- **mat** – *a map* is represented as a nested list (2D matrix). Each element in the list is a row in the matrix. Each position on the map represents a human. The matrix elements are integers.
- **Indices** – the *initial location* of the epidemic is represented as a tuple of two integers: (row, column) where $0 \leq row \leq n, 0 \leq row \leq m$ (n and m are the numbers of rows and columns, respectively).
- **epidemic** – the *epidemic* index is represented as positive integer.

The function updates the map according to the simulation of the epidemic spread.

A matrix cell (i.e., human) that has been infected can spread the disease according to the following rules:

1. The epidemic can spread only to the neighboring cells (up, down, left, right).
2. The epidemic can spread only if another disease does not occupy the neighboring cell. A cell which is not occupied has the value of 0.

Note:

- An occupied cell can be infected in case that the new disease starts in this cell; however, the infection will not spread.
- The function updates the map, and therefore there is no return value.
- Your implantation must be recursive.

Running examples:

```
mat = [[1, 0, 0, 3, 0],
       [0, 0, 2, 3, 0],
       [2, 0, 0, 2, 0],
       [0, 1, 2, 3, 3]]
question3_a(mat, (0,1), 3)
print(mat)
```

output:
[[1, 3, 3, 3, 0],
[3, 3, 2, 3, 0],
[2, 3, 3, 2, 0],
[0, 1, 2, 3, 3]]

```
mat = [[1, 0, 0, 3, 0],
       [0, 0, 2, 3, 0],
       [2, 0, 0, 2, 0],
       [0, 1, 2, 3, 3]]

question3_a(mat, (0,0), 3)
print(mat)
```

Output:
[[3, 0, 0, 3, 0],
[0, 0, 2, 3, 0],
[2, 0, 0, 2, 0],
[0, 1, 2, 3, 3]]

```
mat = [[0,0,0,0],
       [0,0,0,0],
       [0,0,0,0],
       [0,0,0,0]]
question3_a(mat, (0,1), 2)
print(mat)
```

Output:
[[2, 2, 2, 2],
[2, 2, 2, 2],
[2, 2, 2, 2],
[2, 2, 2, 2]]

B. You are asked to implement the function **question3_b(mat)** that returns the number of the largest "healthy community".

You are given as input to the function:

- **mat** – *a map* is represented as a nested list (2D matrix). Each element in the list is a row in the matrix. Each position on the map represents a human. The matrix elements are integers. A healthy human is represented with the value 0 in its matrix entry.

Given a group of n different humans ($1 \leq n \leq$ size of the **mat**) in the matrix **mat**, the group is called a "healthy community" if the group has the following conditions:

1. All the humans in the group are healthy.
2. If $n > 1$: for any two cells in the group, m_1 and m_2, the cells are neighbors (as defined in 3A), or that there are j other humans in the groups, $h_1, h_2, \dots, h_j$ $(1 \leq j \leq n-2)$ such that in the series $m_1, h_1, h_2, \dots, h_j, m_2$, each consecutive elements are neighbors (i.e., m_1 and h_1 are neighbors, h_1 and h_2 are neighbors, and so on).

The second condition can be interpreted as whether there is a "route in the matrix" of healthy humans that connect every two humans in the group.

Note:

- The return value is an integer.
- Your implantation must be recursive.

Running examples:

```
mat = [[1, 0, 0, 3, 0],
       [0, 0, 2, 3, 0],
       [2, 0, 0, 2, 0],
       [0, 1, 2, 3, 3]]
print(question3_b(mat))
```

There are 3 “healthy communities” of sizes 1, 3, and 6. Therefore, the returned value will be 6.