Laboratory Assignment 1

Problem 1 - Indexing `[3 pts]`

Return the first letter on your name as a string.

- You should use your `my_name` variable for correctness.
- `inputs : N/A`
- `output : string`

```
>>>
F
```

Problem 2 - Indexing II `[8 pts]`

Ask for a number from the user to define an index, then return the letter in your name that corresponds to that index starting from 0 (`for input n → return nth index`).

- If the number is negative or the number is more than the number of letters on your name, take modulo and use that as the new number.
- You should use your `my_name` variable for correctness.
- `inputs : input ∈ ℤ`
- `output : string`

```
my_name = "Tarkan"

>>>
Enter a number: 0
T

>>>
Enter a number: -2
a

Explanation: -2 mod 6 = 4, so we took the 4'th indexed character. 6 comes from
the number of letters on my_name.

>>>
Enter a number: 18
T

Explanation: 18 mod 6 = 0, so we took the 0'th indexed character.
```

Problem 3 - Slicing [12 pts]

Ask two numbers from the user to define indexes, then find the substring of your my_name variable taking these two index values as beginning and end. Return this substring.

- Both of these numbers should be taken as modulo of the length of your name as problem 2.
- Order of the numbers does not matter, slice it using minimum - maximum.
- Both indexes are inclusive.
- inputs : input1, input2 ∈ ℤ.
- output : string

```
my_name = "Tarkan"

>>>
Enter first number: 2
Enter second number: 5
rkan

>>>
Enter first number: 0
Enter second number: 6
T

Explanation: index 6 mod 6 = 0, so we slice from 0 th index to 0 th index
(inclusive), which ends up being the first indexed character.

>>>
Enter first number: -2
Enter second number: 3
ka

Explanation: index -2 mod 6 = 4, so we slice from 3 th index to 4 th index
(inclusive), which ends up being the 3rd and 4th indexed characters.

>>>
Enter first number: 3
Enter second number: 9
k

Explanation: index 9 mod 6 = 3, so we slice from 3 th index to 3 th index
(inclusive), which ends up being only the 3rd indexed character.
```

Problem 4 - Vowels [11 pts]

Ask for a string from the user, then count the number of vowels in the input and return the result.

- If the input does not have any vowels, return 0.
- vowels ∈ { 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}
- inputs: input = {x: printable characters except whitespaces and len(x) ∈ [1, 100]}. (This includes letters in English, numbers and punctuation characters.)
- outputs: integer

```
>>>
Enter input: inf211
1

>>>
Enter input: F!ucvh1!UjK\;'an!+!@#
3

Explanation: Only vowels in this string are 'u', 'U' an 'a'.

>>>
Enter input: zxvxsdf
0
```

Problem 5 - Summation [10 pts]

Sum up all the digits on your id number and return the result.

- inputs: N/A
- outputs: integer

my_id = "1234"

```
>>>
10
```

Problem 6 - Factorial [11 pts]

Ask for a number from the user, then calculate and return the factorial of that number.

- Do NOT use any libraries or predefined functions
- inputs: input = {x: x ∈ ℕ and x ≤ 30}
- outputs: integer

```
>>>
Enter input: 4
24
```

```
>>>
Enter input: 12
479001600

>>>
Enter input: 0
1
```

Problem 7 - Divisibility [5 pts]

Ask for a number from the user. If that number is divisible to both 3 and 7 return **True**, else return **False**.

- inputs: input ∈ ℕ
- outputs: boolean

```
>>>
Enter a number: 12
False

>>>
Enter a number: 21
True
```

Problem 8 - Divisibility II [8 pts]

Ask for a number from the user.

1. If that number is only divisible to 3, return 1,
2. if that number is only divisible to 7, return 2,
3. if that number is divisible to both 3 and 7, return 3.

- inputs: input ∈ ℕ
- outputs: integer

```
>>>
Enter a number: 12
1

>>>
Enter a number: 14
2

>>>
Enter a number: 21
3
```

Problem 9 - Prime Number [15 pts]

Ask for a number from the user, then find if that number is a prime number. [1] If the number is a prime number, return **True**, else return **False**.

- `inputs: input = { x: x ∈ ℕ and x > 1}`
- `outputs: boolean`

```
>>>
Enter a number: 12
False

>>>
Enter a number: 23
True
```

Problem 10 - Square root [17 pts]

Find the square root of a given number using Heron's method (Babylonian method) [2] that we discussed in the first lecture.

- Do NOT use any libraries or predefined functions (i.e. `sqrt`)
- Choose an appropriate iteration number.
- Choose an appropriate starting guess. (i.e 1)
- `inputs: input = { x: x ∈ R and 1E9 ≥ x ≥ 0.0}`
- `outputs: float`

```
>>>
Enter a number: 16
4.0

>>>
Enter a number: 12
3.4641016151377544

>>>
Enter a number: 473891.421
688.397720071762
```

[1] https://en.wikipedia.org/wiki/Prime_number
[2] https://en.wikipedia.org/wiki/Methods_of_computing_square_roots#Babylonian_method