Lab session (29th Dec)

1. **Easy**
   1.1. [Caesar Cipher with input] Write a program that declares an character array `str` of size 20. Use `scanf` to read a string into `str` and a positive integer `n`. from the user. Apply Caesar Cipher to shift each character of the string by `n` steps forward in the ASCII table. Store the result in an array `shift_str.` Print the result stored in `shift_str.`
      - Example:
        - Input: HelloWorld4 1
        - Output: IfmmpXpsme5
      - Input: <String>
        <int>
      - Output: <String>

   1.2. [Factorial] Write a function that takes a positive integer n, and returns the factorial of n. Use the input module provided to call your function with n as the argument, and print the value returned by your function followed by a \n.
      - Input: <Int>
        Output: <int>

   1.3. [Repetitions in an array] Write a program that declares an integer array `arr` of size 20. It first takes in a positive integer n<=20 from the user. Then reads n positive numbers and stores them in `arr.` It checks and prints the number of repetitions in `arr.` The result is the sum of the total number of repetitions in the array.
      - Input:
        6
        4 3 4 5 10 12
        Output: 2
      - Input:
        7
        4 3 4 4 3 4 5
        Output: 6
        (which is 4 repetitions of 4 + 2 repetitions of 3).
      - Input: <Int>
        <space separated ints>
      - Output: <Int>

2. **Normal**
   2.1. [Number of letters in the input string] Write a program that inputs two strings `pat` (of length n1<=20) and `str` (of length n2<=20). It creates an array `occur` which stores the number of times `pat[i]` occurs in `str`.
      - The following are sample inputs and outputs
        - Input:

abcd
aabbcc
Output: 2 2 2 0

- Input:
  abcd
  aabbbbbbppcppdpdp
  Output: 2, 6, 1, 2

Input: <String>
<String>
Output: <space separated ints>

2.2. Write a program that reads two positive integers n1 and n2 from the user and prints all prime numbers in [n1, n2].
  - You will have to implement it using a function `is_composite` that takes in an integer m and returns
    - 1 if m is composite
    - 0 otherwise

Input: <Integer>
<Integer>
Output: <space separated ints>

3. Learn by Experiments

3.1. Implement a `my_getline` function to read a line from the user and stores it in an input array `str`. The `my_getline` returns 0 if the operation is successful, and 1 otherwise.
  - `my_getline` should *only* rely on `scanf` for taking a single character as input.