VINUNIVERSITY

Lab Practice № 10: Recursion and Recursive Algorithms

COMP1010 Introduction to Programming **Week 11**

Lab Practice Submission Instructions:

- This is an individual lab practice and will typically be assigned in the laboratory (computer lab).
- Your program should work correctly on all inputs. If there are any specifications about how the program should be written (or how the output should appear), those specifications should be followed.
- Your code and functions/modules should be appropriately commented. However, try to avoid making your code overly busy (e.g., include a comment on every line).
- Variables and functions should have meaningful names, and code should be organized into functions/methods where appropriate.
- Academic honesty is required in all work you submit to be graded. You should **NOT** copy or share your code with other students to avoid plagiarism issues.
- Use the template provided to prepare your solutions.
- You should upload your .py file(s) to the Canvas **before the end of the laboratory session** unless the instructor gave a specified deadline.
- Submit separate .py file for each Lab problem with the following naming format, for example: **V202000999_Lab10.py**. **Note:** If you are working on Jupiter Notebook, you need to download/convert it to Python .py file for submission.
- Late submission of lab practice without an approved extension will incur the following penalties:

  (a) No submission by the deadline will incur 0.25 point deduction for each problem (most of the problems are due at the end of the lab session).

  (b) The instructor will deduct an additional 0.25 point per problem for each day past the deadline.

  (c) The penalty will be deducted until the maximum possible score for the lab practice reaches zero (0%) unless otherwise specified by the instructor.

Problem 1 – Compute the number of *k*-combinations

Write a recursive function called `combination(`n, k`)` to compute the number of k-combinations (without repetition) from a given set S of n elements, denoted as $C(n, k)$. You can use the following recursive formula:

$$C(n, k) = C(n - 1, k - 1) + C(n - 1, k)$$

Prompt the user to input the integer value of n and k, and print out the value of $C(n, k)$ (print 0 if $C(n, k)$ does not exist).

Note: Constraint: your program should be able to run for all $n \leq 30$. Submit your program to CMS. Some examples are followed:

```
Enter two numbers (n, k): 5 1
Combination(5, 1) = 5

Enter two numbers (n, k): 5 3
Combination(5, 3) = 10

Enter two numbers (n, k): 10 3
Combination(10, 3) = 120

Enter two numbers (n, k): 20 5
Combination(20, 5) = 15504

Enter two numbers (n, k): 30 10
Combination(30, 10) = 30045015
```

Problem 2 - Compute the factorial of an positive integer using recursion

Write a recursive function called `fact(`n`)` to compute the factorial of an positive integer n as follows:

$$\texttt{fact}(n) = 1 * 2 * ... * (n-1) * n = n * \texttt{fact}(n-1)$$

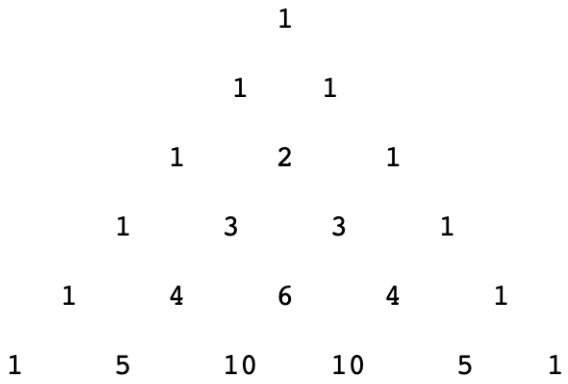
Prompt the user to input a positive integer n, and print out fact(n).

Submit your program to CMS. Sample outputs are as below:

```
Enter an integer: 1
Factorial(1) = 1! = 1

Enter an integer: 3
Factorial(3) = 3! = 6

Enter an integer: 5
Factorial(5) = 5! = 120

Enter an integer: 7
Factorial(7) = 7! = 5040

Enter an integer: 10
Factorial(10) = 10! = 3628800
```

Problem 3 – Generating the Pascal triangle

Write a function which implements the Pascal's triangle:



a) Only using iterative method (CMS is not required).

b) Use recursive method (CMS is required).

Output sample:

```
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
[1, 5, 10, 10, 5, 1]
```

Problem 4 – Producing the Fibonacci numbers out of Pascal's triangle

The Fibonacci numbers are hidden inside of Pascal's triangle. If you sum up the coloured numbers of the following triangle, you will get the 7th Fibonacci number (see Figure 1: Pascal's triangle). See here `https://www.goldennumber.net/pascals-triangle/` how the Fibonacci series is found in Pascal's triangle. Now printing out the first ten Fibonacci numbers line-by-line and submit your solution to CMS. Sample output is shown in Figure 2: Result.

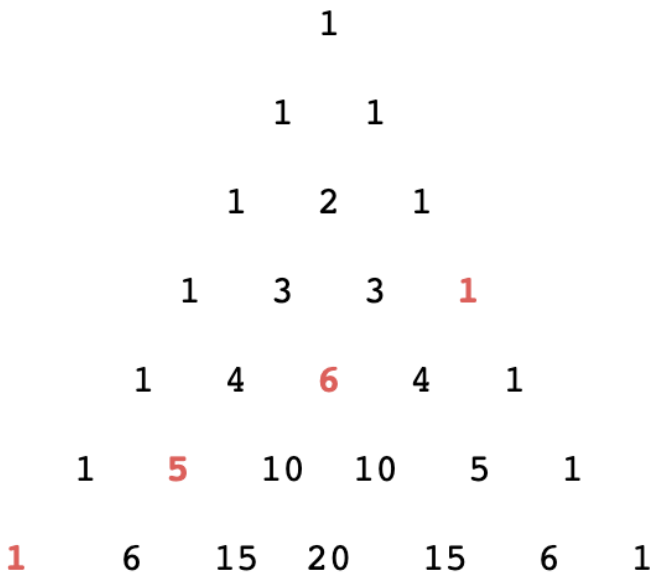


Figure 1: Pascal's triangle

```
1
1
2
3
5
8
13
21
34
```

Figure 2: Result