List no 4

1. Create a class 'FunctionApproximation' in which the components will be the function's argument and accuracy. The class will contain the following methods for calculating the series:

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \quad \text{for all } x$$

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots \quad \text{for all } x$$

$$\tan x = \sum_{n=1}^{\infty} \frac{B_{2n}(-4)^n\left(1-4^n\right)}{(2n)!} x^{2n-1} = x + \frac{x^3}{3} + \frac{2x^5}{15} + \cdots \quad \text{for } |x| < \frac{\pi}{2}$$

$$\sec x = \sum_{n=0}^{\infty} \frac{(-1)^n E_{2n}}{(2n)!} x^{2n} = 1 + \frac{x^2}{2} + \frac{5x^4}{24} + \cdots \quad \text{for } |x| < \frac{\pi}{2}$$

$$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = x + \frac{x^3}{6} + \frac{3x^5}{40} + \cdots \quad \text{for } |x| \le 1$$

$$\begin{aligned} \arccos x &= \frac{\pi}{2} - \arcsin x \\ &= \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} \end{aligned} \quad = \frac{\pi}{2} - x - \frac{x^3}{6} - \frac{3x^5}{40} - \cdots \quad \text{for } |x| \le 1$$

$$\arctan x = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots \quad \text{for } |x| \le 1,\ x \neq \pm i$$

- The program should create an object of type 'FunctionApproximation' and demonstrate the calculations of the mentioned series using the created object.
- The class must have an implemented default constructor (initializing the object's state with default values, e.g., x = 2, n = 100).
- The class must have an overloaded constructor that allows initializing the object's state by passing values to the constructor from the code.
- The 'FunctionApproximation' class is an additional class in the default package - meaning, it's outside the public class that contains the main() method.

Please note that in the context of object-oriented programming, these guidelines describe the necessary elements and functionalities expected within the 'FunctionApproximation' class, to meet the specified requirements.

2. Prepare an alternate solution for task 1. Version 2 of the task's solution requires the creation of additional mathematical functions: a power function and a factorial function for a given number. These functions should be utilized in the process of computing the series values. Additionally, both functions should be defined as static methods within a class in a separate package named 'PackageMath.'

Develop an alternative solution for the first task. In this second version, it is required to create additional mathematical functions: one for exponentiation and another to compute the factorial of a given number. These functions must be used in the calculation process of the series values. Furthermore, both functions should be defined as static methods within a class in a separate package named 'PackageMath.'