**PseudoCode for Assignment 1

```
Function assign1(): #define a function
    Try: $user try so exception can be managed
        a = input()   #Input the user (convert it to an integer as per language), and store it in variable ‘a‘.

        If ‘a‘ is less than 0:
            Print "Please enter a positive integer" or return null

        If ‘a % 2 == 0‘:  # Check if ‘a‘ is even

            Print or return (a * a) ;     #if even compute square of input value

        Else:  # If ‘a‘ is odd

            Print or return (a * a * a ) #if odd compute cube  of input value

    Catch any exception:
        Print "Please provide a valid integer" or return null
End Function
```

**PseudoCode for Assignment 3.1

```
Function factorial(n):
    If n < 0:
        Return "Invalid input, n must be non-negative"
    If n == 0:
        Return 1
    result = 1
    For i from 1 to n:
        result = result * i
    Return result
End Function
```

**PseudoCode for Assignment 3.2

```
Function fibonacci(n):
    If n < 0:
        Return "Invalid input, n must be non-negative"
    If n == 0:
        Return 0
    If n == 1:
        Return 1
    a = 0
    b = 1
    For i from 2 to n:
        temp = a + b
        a = b
        b = temp
    Return b
End Function
```

** Code Modularity **

We break a large program to smaller patches to do a specific task and can be available at any point of time to reuse.

For Ex:
 So there is a situation where we have to get lots of permutations and combinations or need probability then
 we might need factorial of numbers quite often in our program
 if we write factorial logic repeatedly it increases time and space and code is not readable
 so we can modularize that and try to call this function saves our time of development and increase code readability and
 reusability

Benefits:
 Code Reusability
 Increases code Readability
 Easier to fix bugs
 Complexity of code reduces
 development time reduces
 Better Organized
 More Scalable