WIA1002/WIB1002 Data Structure

Tutorial 2: Recursion (Fundamental)

1. Explain the problem that occurs when executing the recursive method f(0).

```
public static int f(int n){
        if (n == 1)
                return n;
        else
                return n * f(n-1);
}
```

The problem that occurs when executing the recursive method f(0) is infinite recursion because n is assigned with the value of 0 where the **base case n == 1** will never be reached/meet. Instead, the value of n keeps decreasing causing endless recursive calls of f. Each recursive call adds a new stack frame which consumes memory. The call stack will grow continuously until it exceeds its limit, triggering a *StackOverflowError* which crashes the program during compilation.

2. Explain the problem that occurs when executing the recursive method f().

```
public static int f(int n) {
        if (n == 0)
                return n;
        else
                return f(n+1) + n;
}
```

- The problem that occurs when executing the recursive method f() is compile-time error because the method f() requires an integer parameter n but the function call does not pass down in an integer argument.
- The problem that occurs when executing the recursive method f(1) is infinite recursion because n is assigned with the value of 1 where the **base case n == 0** will never be reached/meet. Instead, the value of n keeps increasing using endless recursive calls of f. Each recursive call adds a new stack frame which consumes memory. The call stack will grow continuously until it exceeds its limit, triggering a *StackOverflowError* which crashes the program during compilation.

3. Write a recursive method to reverse a string.
   String → gnirts

```
public static String reverse(String str){
    // Handle null case (NullPointerException)
    if (str == null) {
        return null;
    }
    // Check for empty/single-char strings
    if (str.isEmpty() || str.length() <= 1) {
        return str;
    }
    return reverse(str.substring(1)) + str.charAt(0);
}
```

4. Write a recursive method to calculate the following :
   5 + 4 + 3 + 2 + 1.

State the base case and recursive case. Trace your solution using number, N of 5.

Algorithm :

1. Base case = 1
2. Recursive case = n + sum(n-1)

```
public static int sum(int n){
    // Base case = 1
    if (n == 1) {
        return 1;
    }
    // Recursion
    return n + sum(n - 1);
}
```

5. Write a recursive method printDigit that prints an integer argument as its constituent digits, with a blank space separates each digit with the next. For example, if the argument is 4567, this method will print 4 5 6 7 on the screen.

```
/*
    4567 > 10
    printDigit(456)
    sout 7 (4)
    456 > 10
    printDigit (45)
    sout 6 (3)
    45 > 10
    printDigit
    sout 5 (2)
    sout 4 (1)
    */
    // The recursive calls follow a "last in, first out" (LIFO) order, meaning the most recent
        (deepest) call resolves first when the recursion unwinds.
    public static void printDigit(int n){
        // Base case
        if (n < 10) {
            System.out.print(n%10 + " ");
        }else{
            printDigit(n/10);
            System.out.print(n%10 + " ");
        }
}
```