Pre-Practical activity (Memo)

1. Define recursion and how it works

   Recursion is a process in which a function calls itself directly or indirectly to solve a problem. A recursive function typically breaks a problem down into smaller subproblems, solves the smallest version, and uses that solution to solve larger versions of the problem.

2. Explain the concept of tail recursion and how it differs from regular recursion.

   Tail recursion is a form of recursion where the recursive call is the last operation

3. Why is tail recursion generally considered more efficient regarding memory usage than non-tail recursion?

   The last operation is the recursive call therefore the recursive call is immediately returned and consequently no further operations after the recursion.

4. Rewrite this code to be a recursive function:

```
int factorialIterative(int n) {
    int result = 1;
    for (int i = 2; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

   Solution:

```
int factorialRecursive(int n) {
    if (n <= 1)
        return 1;
    else
        return n * factorialRecursive(n - 1);
}
```

5. Rewrite this code to use tail recursion:

```
int factorial(int n) {
    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
}
```

Solution:

```
int factorialTailRecursive(int n, int accumulator = 1) {
    if (n <= 1)
```

6. Write a tail recursive function to compute the factorial of a number.

```
int factorialTailRecursive(int n, int accumulator = 1) {
    if (n <= 1)
        return accumulator;
    else
        return factorialTailRecursive(n - 1, accumulator * n);
}
```

7. Write a recursive function that reverses a string.

```
string reverseStringRecursive(string s) {
    if (s.length() == 0)
        return "";
    else
        return s[s.length() - 1] + reverseStringRecursive(s.substr(0,
s.length() - 1));
}
```