Class Test 7 (Memo)

Determine whether the function is recursive, tail recursive, or iterative.

1)

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

a) Recursive
b) Tail Recursive
c) Iterative
d) None of the above (1)

2)

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

a) Recursive
b) Tail Recursive
c) Iterative
d) None of the above (1)

3)

```
int sum(int arr[], int n) {
    int total = 0;
    for (int i = 0; i < n; i++) {
        total += arr[i];
    }
    return total;
}
```

a) Recursive
b) Tail Recursive
c) Iterative
d) None of the above (1)

4)

```
int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

a) Recursive
b) Tail Recursive
c) Iterative
d) None of the above (1)

5)

```
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
```

a) Recursive
b) Tail Recursive
c) Iterative
d) None of the above (1)

6)

```
int power(int base, int exp) {
    int result = 1;
    while (exp > 0) {
        result *= base;
        exp--;
    }
    return result;
}
```

a) Recursive
b) Tail Recursive
c) Iterative
d) None of the above (1)

7)

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

a) Recursive
b) Tail Recursive
c) Iterative
d) None of the above (1)

8)

```
int sumTailRecursive(int arr[], int n, int total = 0) {
    if (n == 0)
        return total;
    return sumTailRecursive(arr, n - 1, total + arr[n - 1]);
}
```

a) Recursive
b) Tail Recursive
c) Iterative
d) None of the above (1)

9) Rewrite the following function as a recursive function.

```
int sumIterative(int arr[], int n) {
    int total = 0;
    for (int i = 0; i < n; i++) {
        total += arr[i];
    }
    return total;
}
```

(3)

Solution:

```
int sumRecursive(int arr[], int n) {
    if (n == 0)
        return 0;
    return arr[n - 1] + sumRecursive(arr, n - 1);
}
```

10) Rewrite the following function using tail recursion.

```
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
```

(3)

Solution:

```
int gcdTailRecursive(int a, int b) {
    if (b == 0)
        return a;
    return gcdTailRecursive(b, a % b);
}
```

11) Rewrite the following function iteratively.

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

(3)

Solution:

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

12) The following recursive function, determine if the function works correctly. If it does not explain the issue and rewrite the function to fix it.

```
int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 3); }
```

(3)

Solution:

```
int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```