CS2023 - Data Structures and Algorithms
Take Home Assignment

Week 3 - Recursion

By: Sajeev Kugarajah (210554M)

Question 1

Please write pseudo codes to solve the following problems.
Note: Your solutions must be recursive

1. Calculating the n^{th} factorial of a number.

```
FUNCTION Factorial(n):
    IF n = 0 THEN:
        RETURN 1
    ELSE:
        RETURN n*Factorial(n-1)
    END IF

BEGIN
INPUT num
fact <- Factorial(num)
OUTPUT fact
END
```

2. Searching for an element in an array.

```
FUNCTION Linear_Search(Array, Item, Index=0):
    IF Index >= size of Array:
        RETURN -1
    ELSE IF Array[Index] == Item:
        RETURN Index
    ELSE:
        Linear_Search(Array, Item, Index+1)
    END IF

BEGIN
INPUT array,item
index <- Linear_Search(array, item)
OUTPUT index
END
```

3. Checking whether a given string is a palindrome.

```
FUNCTION Pallindrome_Checker(String, FirstIndex, LastIndex):
    IF LastIndex – FirstIndex <= 0:
        RETURN TRUE
    ELSE IF String[FirstIndex] == String[LastIndex]:
        RETURN Pallindrome_Checker(String, FirstIndex+1, LastIndex-1)
    ELSE:
        RETURN FALSE
    END IF

BEGIN
INPUT string
size <- string size
is_pallindrome <- Pallindrome_Checker(string, 0, size-1)
OUTPUT is_pallindrome
END
```

Question 2

Comment on the time complexities of the above algorithms.
Note: Just stating the complexities is fine, no need to explain

1. T(n) = O(n)
2. a) worst case T(n) = O(n)
   b) best case T(n) = O(1)
   c) average case T(n) = O(n)
3. a) worst case T(n) = O(n)
   b) best case T(n) = O(1)
   c) average case T(n) = O(n)

Question 3

Analyze the worst case time complexity of the Merge algorithm.
Note: Do not just state the worst case complexity, you are required to explain how you arrived at the answer.

Line No	Code	Cost	Times
1	MERGE(A, p, q, r)	T(n)	1
2	n1 ← q – p + 1	C1	1
3	n2 ← r – q	C2	1
4		-	-
5	for i ← 0 to n1-1	C3	n1 + 1
6	L[i] ← A[p+i]	C4	n1
7	for j ← 0 to n2-1	C5	n2 + 1
8	R[j] ← A[(q+1)+j]	C6	n2
9	L[n1] ← infinity	C7	1
10	R[n2] ← infinity	C8	1
11	i ← 0	C9	1
12	j ← 0	C10	1
13	for k ← p to r	C11	n + 1
14	if L[i] ≤ R[j]	C12	n
15	A[k] ← L[i]	C13	n/2
16	I ← i+1	C14	n/2
17	else		
18	A[k] ← R[j]	C15	n/2
19	j ← j+1	C16	n/2

$T(n) = C1 + C2 + C3(n1 + 1) + C4.n1 + C5(n2 + 1) + C6.n2 + C7 + C8 + C9 + C10 + C11(n + 1) + C12.n + (C13 + C14 + C15 + C16).n/2$

$T(n) = (C1 + C2 + C3 + C5 + C7 + C8 + C9 + C10 + C11) + C3.n1 + C4.n1 + C5.n2 + C6.n2 + C11.n + C12.n + (C13 + C14 + C15 + C16).n/2$

$C3 = C5 \mid C4 = C6 \mid C13 = C15 \mid C14 = C16 \mid n1 + n2 = n$

$T(n) = (C1 + C2 + 2.C3 + C7 + C8 + C9 + C10 + C11) + C3(n1 + n2) + C4(n1 + n2) + C11.n + C12.n + (2.C13 + 2.C14).n/2$

$T(n) = (C1 + C2 + 2.C3 + C7 + C8 + C9 + C10 + C11) + C3.n + C4.n + C11.n + C12.n + (C13 + C14).n$

$T(n) = (C1 + C2 + 2.C3 + C7 + C8 + C9 + C10 + C11) + (C3 + C4 + C11 + C12 + C13 + C14).n$

$T(n) = O(n)$