Philippe Carrier 40153985

Assignment 2 – Programming Question

Calculator 1

```
algorithm calculator(equation):
    Input the sequence of the equation in a string
    Output the result of the equation

    operators = Stack()
    numbers = Stack()
    for each char in equation do
        if char in operators then
            if char == ‘(’ then
                operators.push(char)
            else if char == ‘)’ then
                while operators.top() != ‘(‘ do
                    y = numbers.pop()
                    x = numbers.pop()
                    equ = operators.pop()
                    numbers.push(x equ y)
                operators.pop(operators.top())
            else
                while operators.top() >= char do
                    y = numbers.pop()
                    x = numbers.pop()
                    equ = operators.pop()
                    numbers.push(x equ y)
                operators.push(char)
        else
            numbers.push(char)
    while not operators.isEmpty() do
        y = numbers.pop()
        x = numbers.pop()
        equ = operators.pop()
        numbers.push(x equ y)
    return numbers.pop()
```

Calculator 2

```
algorithm calculator(equation):
    Input the sequence of the equation in a string
    Output the result of the equation

    result = 0
    for i=0 to equation.length - 1 do
        switch(equation[i])
            case “)”:
                sum = calculator(equation.substring(i + 1))
            case “)”:
                return sum
            case “!”:
                if equation[i + 1] == “=” then
                    return result != calculator(equation.substring(i + 1))
                else
                    result = factorial(result)
            case “^”:
                result = result ^ (next number)
            case “-_”:
                result = - (next number)
            case “+”:
            case “-”:
            case “*”:
            case “/”:
                if next operator > current operator then
                    result = result  operator calculator(equation.substring(i + 1))
                else
                    result = result operator (next number)
            case “>”:
            case “>=”:
            case “<”:
            case “<=”:
                return result operator calculator(equation.substring(i+1))
    return result
```

a) The time complexity of the first calculator pseudocode is O(n) since the you have to iterate through the equation. The rest of the operation in the first calculator are O(1). The space complexity is O(n) since in the worst case, the total space occupy by both stack will be n space. As for the second calculator, the pseudocode time complexity is O(n^2). Once again, the equation is iterated over (O(n)) and there is also the substring method which is O(n). The space complexity is also O(n^2). The call stack is growing with each call to the method (recursive) and there is also the substring method which creates variable length of string.

b) The second version of the calculator is a multiple recursive method since each call to the method can call multiple the recursive method. This type of recursion can have an impact on the time and memory complexity since the same calculation can be done by multiple recursive call which lengthen the time of the method and space as well.

c)

Calculator 1 log:

```
5 == 4
false
5 + 9 >= 12 / 4
true
(30 - 4) / 2 != 13 + 0
false
5+4
9.0
(5+4)
9.0
(5+4*3)/2
8.5
(5!+4*3)/2
66.0
(-2+4*5)/(10+1)-9/3
-1.3636363636363635
(4+20/5)^3-50
462.0
(-2+4*5)+(10+1)/(9/3)
21.666666666666668
(-2+4*5)/(10+1)-9/3
-1.3636363636363635
((5!/4-10)*2+3)*5
215.0
-52-98/10*(-94)
869.2
-15/-71+46-(-57*47)--93/(-21--26)+-20
2723.811267605634
25/(2-43)+-31/(88+74-8+-46/(79*-22))
-0.8110202086114877
-44 / 9! / 7 - (71 / 52 + 3) * 31 / 65 - 89 + -1 / -54
-91.06345146594688
(((8+2*3)!-7 + 3/-2) + 40)/(115 - 20)
9.176662234894737E8
((8+-2*3!+21+ 14/-3) + 92)/(5! - 6*7)
1.3376068376068375
```

```
93/5*(4 + 18 - 3^3)
-93.0
8^8 / 10! + 52 / (14! - 8)
4.623350970614115
5^9/40 - 75*4^8
-4866371.875
52/5*12*(7 - 24*-5)/32
495.30000000000007
581/(5! - 72 /4) + 908 / (80 -3^3)
22.82815390307066
```

Calculator 2 logs:

```
5==4
false
5+9>=12/4
true
(30-4)/2!=13+0
false
5+4
9.0
(5+4)
9.0
(5+4*3)/2
8.5
(5!+4*3)/2
66.0
(-2+4*5)/(10+1)-9/3
-1.3636363636363635
(4+20/5)^3-50
462.0
(-2+4*5)+(10+1)/(9/3)
21.666666666666668
((5!/4-10)*2+3)*5
215.0
-52-98/10*(-94)
869.2
-15/(-71)+46-(-57*47)-(-93)/(-21-(-26))+(-20)
2719.011267605634
25/(2-43)+(-31)/(88+74-8+(-46)/(79*(-22)))
-0.8110202086114877
-44/9!/7-(71/52+3)*31/65-89+(-1)/(-54)
-91.06345146594688
(((8+2*3)!-7+3/(-2))+40)/(115-20)
9.176662234894737E8
```

```
((8+(-2)*3!+21+14/(-3))+92)/(5!-6*7)
1.3376068376068375
93/5*(4+18-3^3)
-93.0
8^8/10!+52/(14!-8)
4.623350970614115
5^9/40-75*4^8
-4866371.875
52/5*12*(7-24*(-5))/32
1341.6000000000001
581/(5!-72/4)+908/(80-3^3)
22.82815390307066
```