Task 3

Made by: Hamza Ayman

**Recursion:**

Recursion is a method of solving a problem by breaking it down into smaller problems of the same type. The recursive solution to a problem is typically expressed in terms of a function that calls itself. The function must have a base case, which is a simple case that can be solved directly. The function must also have a recursive case, which breaks the problem down into smaller subproblems. The recursive case is then solved by calling the function again.

For example, the factorial function can be defined recursively as follows:

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

This function calls itself recursively to calculate the factorial of n - 1. The base case is when n is 0, in which case the function returns 1.

**Tabulation:**

Tabulation is an iterative method of solving a problem by breaking it down into a table of values. The table is created by filling in the values of the problem one by one. The table can then be used to solve the problem.

For example, the Fibonacci sequence can be generated using tabulation as follows:

def fibonacci(n):

table = [[0 for i in range(n)] for j in range(n)]

table[0][0] = 0

table[1][0] = 1

for i in range(2, n):

table[i][0] = table[i - 1][0] + table[i - 2][0]

return table[n - 1][0]

This function creates a table of 2 \* n values. The first row of the table contains the Fibonacci numbers up to n - 1. The second row of the table contains the Fibonacci numbers up to n. The function then returns the value of the Fibonacci number at index n - 1 in the second row of the table.

**Memoization:**

Memoization is a technique for improving the performance of recursive functions by storing the results of previously computed subproblems. This is done by creating a table of the results of all possible subproblems. When a subproblem is encountered, the table is checked to see if the result has already been computed. If it has, the result is retrieved from the table. If it has not, the subproblem is computed and the result is stored in the table.

For example, the factorial function can be memoized as follows:

def factorial(n):

table = {}

def helper(n):

if n in table:

return table[n]

else:

result = n \* helper(n - 1)

table[n] = result

return result

return helper(n)

This function creates a table that maps from integers to their factorials. The helper function checks the table to see if the factorial of n has already been computed. If it has, the helper function returns the value from the table. If it has not, the helper function computes the factorial of n and stores it in the table.

**Processors of iPhone:**

The iPhone uses a variety of processors, depending on the model. The original iPhone used the Samsung S3C6410A 32-bit ARMv6 processor. The iPhone 3G and iPhone 3GS used the Apple A4 32-bit ARMv7 processor. The iPhone 4, iPhone 4S, and iPhone 5 used the Apple A5 32-bit ARMv7 processor. The iPhone 5S, iPhone 5C, and iPhone 6 used the Apple A6 64-bit ARMv8 processor. The iPhone 6S, iPhone 6S Plus, iPhone SE (1st generation), and iPhone 7 used the Apple A9 64-bit ARMv8 processor. The iPhone 7 Plus, iPhone 8, iPhone 8 Plus, and iPhone X used the Apple A11 Bionic 64-bit ARMv8.3 processor. The iPhone XS, iPhone XS Max, and iPhone XR used the Apple A12 Bionic 64-bit ARMv8.3 processor. The iPhone 11, iPhone 11 Pro, and iPhone 11 Pro Max used the Apple A13 Bionic 64-bit ARMv8.4 processor. The iPhone 12, iPhone 12 mini, iPhone 12 Pro, and iPhone 12 Pro Max used the Apple A14 Bionic 64-