Pseudocode Challenges

1. Calculate the sum of an array:

Function calculate\_array\_sum(array):

Integer total = 0

Integer Length = number of items in array

While length != 0:

Total = total + array[length]

Length = length – 1

Return total

1. Find the maximum value:

Function maximum value(list):

Length = number of items in list

Integer max\_value = list[length]

While length != 0:

If list[length] > max\_value:

max\_value = list[length]

return max\_value

1. Count vowels in a string:

Function count\_vowels(string):

Integer total = 0

Integer length = number of items in string

While length != 0:

If list[length] == a or == e or == I or == o or == u:

Total = total + 1

Length = length – 1

1. Reverse string:

Function reverse\_string(string):

Integer Length = length of string

String reverse\_string

Integer counter = 1

While length != 0:

reverse\_string[counter] = string[length]

counter = counter - 1

length = length – 1

return reverse\_string

1. Check for prime number:

Function check\_prime(number):

Integer factors = 0

Integer counter = 1

While counter != number:

If number % counter != 0:

Factors = factors + 1

If factors > 2:

Output “Number is not a prime”

Else:

Output “Number is a prime”

1. Sort an array:

Function sort\_array(array):

Integer counter = 1

Length = length of array

While counter != length of array:

If array[counter] > array[counter + 1]:

Swap numbers in array

Counter = counter + 1

1. Factorial:

Function factorial(number):

Integer counter = number -1

While counter != 0:

Number = number \* counter

Counter = counter – 1

Return counter

1. Fibonacci sequence:

Function Fibonacci(n):

Integer number

List sequence

Sequence[1] = 1

Integer counter = 2

While counter != n:

Number = sequence[counter - 1] + sequence[counter - 2]

Sequence[counter] = number

Counter = counter – 1

Return sequence

1. Palindrome:

Function palindrome(word)

Integer length = length of word

Integer Counter = 1

List reversed\_word = “”

while length != 0:

reversed\_word[counter] = word[length]

counter = counter + 1

length = length – 1

if reversed\_word == word:

output “Word is a palindrome”

else:

output “Word is not a palindrome”

1. Find GCD:

Function find\_GCD(number\_1, number\_2):

Integer number = 1

Integer GCD = 1

While number != number\_1 and != number\_2:

If number % number\_1 == 0 and number % number\_2 == 0:

GCD = number

Number = number + 1

Return GCD

1. Matrix addition:

Function matrix\_addition(matrix\_1 ,matrix\_2):

Integer added\_matrix 2 dimensions

If matrix\_1 width > matrix\_2 width:

Integer width = matrix\_1 width

Else:

Integer width = matrix\_2 width

If matrix\_1 height > matrix\_2 height:

Integer height = matrix\_1 height

Else:

Integer height = matrix\_2 height

Integer counter = 1

While counter != height:

Integer counter\_2 = 1

While counter\_2 != width:

Added\_matrix[counter][counter\_2] = matrix\_1[counter][counter\_2] + matrix\_2[counter][counter\_2]

Counter\_2 = counter\_2 + 1

Counter = counter + 1

Return added\_matrix

1. Count the words in a sentence:

Function count\_words(sentence):

Integer counter = 1

Integer n = 1

While counter != sentence length:

If sentence[counter] == “ ”:

n = n + 1

counter = counter + 1

return n