Assignment by OME JOHN

Algorithm to find the Fibonacci sequence.

Fibonacci series is defined as a sequence of numbers in which the first two numbers are 1 and 1 or 0 and 1, depending on the selected beginning point of the sequence, and each subsequent number is the sum of the previous two.

* Start
* Declare variables n1, n2, count, nth
* Assign the variables n1=0, n2=1, count=0
* Enter the number of terms of Fibonacci series to be printed
* If the number of terms entered is less than or equal to zero, ask the user to enter a positive integer.
* If the number of terms entered is equal to 1, return n2 to the user.
* If the user enters a correct number of terms, initiate a loop.
* While count is less than no of terms
* Print out n1
* Assign the variable nth to the sum of n1 and now
* Update n1 to equal n2
* Update n2 to equal nth
* Increase the value of count each time by 1
* Print the value of count
* End

Algorithm to sort a given set of numbers (Bubble sort)

It is an in-place sorting algorithm that finds maximum element in each cycle and puts it in appropriate position in list by performing swapping adjacent elements until they are in correct order.

* Start
* Ask the user to enter a given items of array using input
* Take the first element, compare the current element with the next element
* If the current element is greater than the next element, swap them.
* If the current element is less than the next element, then move to the next element
* Repeat step 3 to step 5 until all elements are sorted.
* Stop

Algorithm to convert from one base to another

* Start
* Create a string variable called digits to store numbers from 0 to 9 and letter a to d
* Initiate a function with two parameters called decimal numbers and base
* Define an empty list and store in a variable called remainder stack, to store the remainder of the modulus division
* Initiate a loop
* While decimal number is greater than 0
* decimal number % base
* Store the result in a variable called remainder
* Add the remainder to remainder stack
* Get the floor division between decimal number and the base
* Store the result in decimal number
* End loop
* Create a new variable to store the new digits called newdigits
* Initiate a loop
* While remainder stack
* Remove each item from remainder stack and index into digits
* Store the result in a variable index digits
* Add the variable index digits to the variable newdigits
* Return newdigits
* Call the name of the function
* End

Algorithm to find the factorial of a number

* Start
* Enter a positive integer number Num
* Declare an integer variable – Factorial and give it an initial value of 1.
* Check that user entered integer number and prompt them to try again if not.
* Initiate a loop
* While Num is greater than zero
* Multiply Factorial by Num and store result in Factorial
* Subtract 1 from Num
* End While Loop
* Print out Factorial
* Stop

Algorithm to find the sum of two integer numbers

* Start
* Create two variable num1 and num2 to take the input of first and second number respectively
* If a non-integer number is entered by the user, print number must be an integer
* Create a variable add to store the result of the sum of num1 and num2
* Print add
* Stop