Algorithm & Pseudocode Questions

Q1. Sum of Two Numbers

Problem:
Write an algorithm to find the sum of two numbers provided by the user.

Algorithm:

1. Start
2. Declare two variables `num1` and `num2`
3. Input the values for `num1` and `num2` from the user
4. Add `num1` and `num2`, and store the result in a variable `sum`
5. Output `sum`
6. End

Pseudocode:

```
Start
  Declare num1, num2, sum
  Input num1
  Input num2
  sum = num1 + num2
  Output sum
End
```

Q2. Find the Largest Number

Problem:
Given a list of numbers, design an algorithm to find the largest number in the list.

Algorithm:

1. Start
2. Initialize a variable `max` to the first element in the list
3. Traverse each element `num` in the list:
  - If `num` is greater than `max`, set `max = num`
4. Output `max`
5. End

Pseudocode:

```
Start
  Declare list of numbers
  Set max = list[0]
  For each num in list:
    If num > max:
      max = num
  Output max
End
```

Q3. Swap Two Numbers

Problem:
Write an algorithm to swap the values of two variables.

Algorithm:
1. Start
2. Declare two variables `a` and `b`
3. Input values for `a` and `b`
4. Use a temporary variable `temp` to swap:
  - temp = a
  - a = b
  - b = temp
5. Output the swapped values of `a` and `b`
6. End

Pseudocode:
```
Start
   Declare a, b, temp
   Input a
   Input b
   temp = a
   a = b
   b = temp
   Output a, b
End
```

Q4. Count Down from a Number
Problem:
Create an algorithm that takes an input number and counts down to 0, printing each number.

Algorithm:
1. Start
2. Input a number `n`
3. While `n` is greater than or equal to 0:
  - Output `n`
  - Decrement `n` by 1
4. End

Pseudocode:
```
Start
   Declare n
   Input n
   While n >= 0:
      Output n
      n = n - 1
End
```

Q5. Check Even or Odd
Problem:
Design an algorithm to determine if a given number is even or odd.

Algorithm:
1. Start
2. Input a number `n`
3. If `n % 2 == 0`, output "Even"

4. Otherwise, output "Odd"
5. End

Pseudocode:

```
Start
   Declare n
   Input n
   If n % 2 == 0:
      Output "Even"
   Else:
      Output "Odd"
End
```

Q6. Reverse a List

Problem:
Write an algorithm to reverse the elements in a list.

Algorithm:

1. Start
2. Input a list of elements
3. Initialize an empty list `reversedList`
4. Traverse the original list from the end to the beginning:
  - Append each element to `reversedList`
5. Output `reversedList`
6. End

Pseudocode:

```
Start
   Declare list, reversedList
   For i from length of list - 1 to 0:
      Append list[i] to reversedList
   Output reversedList
End
```

Q7. Find the Smallest Number

Problem:
Design an algorithm to find the smallest number in a list of numbers.

Algorithm:

1. Start
2. Initialize a variable `min` to the first element in the list
3. Traverse each element `num` in the list:
  - If `num` is less than `min`, set `min = num`
4. Output `min`
5. End

Pseudocode:

```
Start
   Declare list of numbers
   Set min = list[0]
   For each num in list:
      If num < min:
         min = num
   Output min
End
```

Q8. Calculate Factorial

Problem:
Create an algorithm that computes the factorial of a given number.

Algorithm:

1. Start
2. Input a number `n`
3. Initialize a variable `factorial = 1`
4. For each `i` from 1 to `n`:
  - Multiply `factorial` by `i`
5. Output `factorial`
6. End

Pseudocode:

```
Start
   Declare n, factorial = 1
   Input n
   For i from 1 to n:
      factorial = factorial * i
   Output factorial
End
```

Q9. Check for Prime Number

Problem:
Write an algorithm to check if a given number is a prime number.

Algorithm:

1. Start
2. Input a number `n`
3. If `n` is less than 2, output "Not Prime"
4. For each `i` from 2 to the square root of `n`:
  - If `n % i == 0`, output "Not Prime" and exit
5. If no divisors were found, output "Prime"
6. End

Pseudocode:

```
Start
   Declare n
   Input n
   If n < 2:
      Output "Not Prime"
   For i from 2 to sqrt(n):
      If n % i == 0:
         Output "Not Prime"
         Exit
   Output "Prime"
End
```

Q10. Bubble Sort

Problem:
Implement the bubble sort algorithm to sort a list of numbers in ascending order.

Algorithm:

```
1. Start
2. Input a list of numbers
3. Repeat the following for each element in the list:
  - Traverse the list and compare adjacent elements
  - If the element is greater than the next, swap them
4. Repeat this process until the list is sorted
5. Output the sorted list
6. End
```

Pseudocode:

```
Start
   Declare list
   For i from 0 to length of list - 1:
      For j from 0 to length of list - i - 1:
         If list[j] > list[j+1]:
            Swap list[j] and list[j+1]
   Output list
End
```