1 What is Pseudocode?

Pseudocode is, essentially, non-executable code. The idea is to express an algorithm in language that is more precise than spoken word (e.g. American Standard English) but to not weigh yourself down with technical details that are not related to the overall algorithm design.

For example, in order to compute the first n Fibonacci numbers and store them in an array in the C programming language, then print them, one might write something like:

```c
#include <stdio.h>
#include <stdlib.h>

void fib(unsigned int[], unsigned int);

int main(int argc, char* argv[])
{
        if (argc == 1) {
                printf("Need an integer argument\n");
                return EXIT_FAILURE;
        }
        unsigned int n = atoi(argv[1]);
        unsigned int* fib_arr = malloc(n * sizeof(*fib_arr));
        fib(fib_arr, n);
        for (unsigned int i = 0; i < n; i++) {
                printf("The %d-th Fibonacci Number is %d\n", i, fib_arr[i]);
        }
        return EXIT_SUCCESS;
}

void fib(unsigned int fib_arr[], unsigned int n)
// Assume array fib_arr is already allocated on the heap
{
        fib_arr[0] = 0;
        if (n > 0) {
                fib_arr[1] = 1;
        }
        for (unsigned int i = 2; i < n; i++) {
                fib_arr[i] = fib_arr[i - 1] + fib_arr[i - 2];
        }
}
```

Notice how much superfluous information is here, at least with regards to what the algorithm actually does. There is a function prototype, a `malloc` call that serves only to acquire memory on the heap, there is a check of the `argc` value, etc. This code also has problems, such as not checking the return value of `fib_arr` (to see if memory was successfully allocated) and not considering what happens when n does not fit within a single word in memory (or what happens when the Fibonacci Numbers get too large), and this can distract from the overall idea of the algorithm.

On the other hand, the pseudocode for this might look something like:

Algorithm 1 Stores the first n Fibonacci Numbers in an array and prints them. Assume $n \geq 1$.

1: **function** FIB(n)
2: `fib-arr` is an empty array of length n
3: `fib-arr`$[0] = 0$
4: `fib-arr`$[1] = 1$
5: **for** $i = 2$ **to** n **do**
6: `fib-arr`$[i] =$ `fib-arr`$[i-1] +$ `fib-arr`$[i-2]$
7: PRINT(`fib-arr`)

Notice that this pseudocode captures the idea of the algorithm without getting bogged down with unnecessary details. How memory is allocated, how to handle arbitrary-precision arithmetic, etc., are not relevant to this algorithm.