```
ADD-BITS(A, B)
    n = MAX(A.length, B.length) + 1
    C = ARRAY(INTEGER, n)
    carry = 0
    i = C.length
    j = A.length
    k = B.length
    while i <= n
        if j > 0 and k > 0
            C[i] = (A[j] + B[k] + carry) mod 2 + carry
            carry = (A[j] + B[k] + carry) / 2
        else if j == 0 and k == 0
            C[i] = carry
        else if j = 0
            C[i] = (B[k] + carry) mod 2
            carry = (B[k] + carry) / 2
        else
            C[i] = (A[j] + carry) mod 2
            carry = (A[j] + carry) / 2
        if j == 0
            j = j - 1
        if k == 0
            k = k - 1
        i = i - 1
```

Loop invariant: at the start of every iteration, C[i-1 .. n] will always have the sum of bits of A[j-1 .. A.length] and B[k-1 .. B.length]

Initialization: C[n-1 .. n] is the null array, therefore it contains the sum of A[A.length-1, A.length]=null and B[B.length-1 .. B.length]=null
Maintenance: Suppose C[i .. n] contains the sum of B[k .. B.length] and A[j-1 .. A.length] and the carry, if j and k are still greater than the lower bounds of both A and B then C[i] = (B[k] + A[j] + carry) mod 2, which means that C[i] will hold the sum of the tow plus whatever carry out there from the previous operation. If we already summed either of all elements of A, or B or both, C[i] will take the carry, as if A[j] or B[k] or both of them are zero.
Therefore C[i] will hold the result of the sum of A[j] and B[k] plus the carry.
Therefore C[i-1 .. n] will contain the sum of B[k-1 .. B.length] and A[j-1 .. A.length] and the carry
Termination: On termination we have i = 0, k = 0, j = 0 , therefore C[1 .. n] contains the sum of B[1 .. B.length] and A[1 .. A.length].