Proof: The proof is by the method of ~~strong~~ mathematical induction on n.

Base Case: n=0. The function returns 0 in line 4, which is correct since the 0th Fibonacci no. is 0.
n=1. The function returns 1 in line 6, which is correct since the 1st Fibonacci number is 1.

Induction hypothesis: For all n≤K, n∈N, recursive_nthFibonacci(n) correctly computes the nth Fibonacci number.

Induction Step: Let, n=K+1. The function evaluates the aft else block in line 7. In line 8, recursive_nthFibonacci(K+1) = recursive_nth Fibonacci((K+1)-1) + recursive_nthFibonacci((K+1)-2 = K-1).
= K
By induction hyp, both recursive_nthFibonacci(K) and recursive_nthFibonacci (K-1) is computed correctly.
∴ recursive_nthFibonacci(K+1) is computed correctly, since we have implemented the correct recursive function call for the nth Fibonacci number □

```
1. int iterative_nthFibonacci(int n)
2. {
3.   int f_i=0;
4.   if(n==0)
5.     return f_i;
6.   int f_j, f_K;
7.   f_j=1;
8.   if(n==1)
9.     return f_j;
10.  for(int i=2; i<=n; i++)
11.  {
12.    f_K=f_i+f_j;
13.    f_i=f_j;
14.    f_j=f_K;
15.  }return f_K; }
```

Loop Invariant: At the start of each iteration with index i, f_i = F(i-2) and f_j = F(i-1)

Initialization: Before the 1st iteration of the loop, we initialize f_i to 0 in line 3 and f_j to 1 in line 7. Initially, index i is initialized to 2.
∴ F(i-1) = F(1) = 1. and F(i-2) = F(0) = 0.
We know these are True and we have respectively initialized f_j to 1 and f_i to 0. ∴ Loop invariant holds.