**Problem 2**

a. False:

b. True:

c. True:

**Problem 4:**

**int** fibonacci(**int** n) {

**int** f0 = 0;

**int** f1 = 1;

**if** (n == 0) {

**return** f0;

}

**if** (n == 1) {

**return** f1;

}

**for** (**int** i = 1; i < n; i++) {

f1 = f0 + f1;

f0 = f1 - f0;

}

**return** f1;

}

The running time is: O(n)

Proof:

* The base case: n = 0 or n = 1, it is correct
* For case k: f1 is computed to new value equal to previous f0, f1 and f0 receive previous value of f1, the assignment running k times and f(k) = f1
* For case k = n: f(n) = f1

**Problem 6:** Count 0 and 1

Input: a length-n array A consisting of 0s and 1s, arranged in sorted order

Output: total number of 0s and 1s in the array

Array count\_0\_1(Array arr) {

c0 = 0

c1 = 0

size <- length of arr

i <- 0

for i = 0 to size do {

if arr[i] == 1 then {

break

}

}

c0 <- i

If (i != size) {

c1 <- size – i

}

return { c0, c1 };

}

**Proof:**

|  |  |
| --- | --- |
| Algorithm | Operations |
| Array count\_0\_1(Array arr) {  c0 = 0  c1 = 0  size <- length of arr  i <- 0  for i = 0 to size do {  if arr[i] == 1 then {  break  }  }  c0 <- i  If (i != size) {  c1 <- size – i  }  return { c0, c1 };  } |  |
| 1 |
| 1 |
| 1 |
| 1 |
|  |
| n |
| 2 |
| 1 |
|  |
|  |
|  |
| 1 |
| 1 |
| 2 |
|  |
|  |
| 2 |
|  |
|  |
| **Summary:** Operations = n + 13 🡪 Algorithm runs in O(n) time | |