

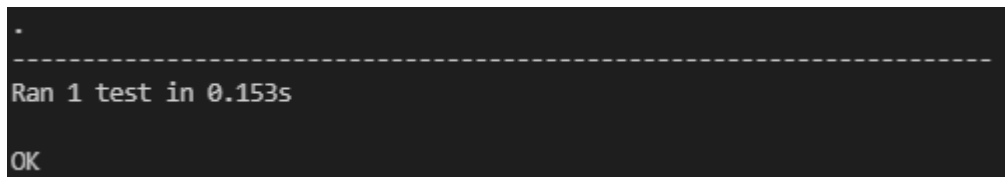
## Some Info.

### b] Missing Invariant and Program Verification:

The Program can be found in '[a3/wlang/q4b.prg](#)'

- I've also pasted it below for your reference.

```
havoc x, y;
assume (y >= 0);
c := 0;
r := x;
while c < y
  inv (c <= y and r = x + c)
do {
  r := r + 1;
  c := c + 1
};
assert (r = x + y)
```



A screenshot of a terminal window with a dark background. It shows a single line of output: "Ran 1 test in 0.153s". Below this line, the text "OK" is visible, indicating that the test passed without any assertion errors.

As we can see, no assertion errors were raised. Hence the invariant correctly verifies the program and is good enough to prove the post-condition.

### c] Branch Coverage:

The following statements have not been covered for the stated reasons:

#### i] sym.py:

|                                |  |
|--------------------------------|--|
| sym.py:437<br>to<br>sym.py:467 | The 2 methods covering these lines are meant for – when the program is run explicitly, and they take the filename (of input file) as an argument. <code>_parse_args()</code> throws an error when called explicitly, since no argument could be passed. Although we can create subprocesses to run them from our unit tests, but still, they will not show up in the coverage (since we would be running them as subprocess and not the same process). |
|--------------------------------|--|

## d] A different program and its Verification:

The Program can be found in '[a3/wlang/q4d.prg](#)'

The program computes  $(x^2 - y^2)$  using its equivalent form  $(x - y) * (x + y)$ ; such that  $x$  &  $y$  are positive integers.

The loop calculates  $(x-y) * (x+y)$  by incrementing, decrementing, and multiplication.

- I've also pasted it below for your reference.

```
havoc x, y;
assume (x >= 0 and y >= 0);

r1 := x;
r2 := x;
c := y;
res := r1 * r2;

while c > 0
  inv (c >= 0 and (r1 = x + (y - c) and (r2 = x - (y - c) and res = (x*x)-((y-c) * (y-c))))
  do {
    r1 := r1 + 1;
    r2 := r2 - 1;
    res := r1 * r2;

    c := c - 1
  };

assert (res = x*x - y*y)
```



```
*
-----
Ran 1 test in 0.618s
OK
```

As we can see, no assertion errors were raised. Hence the invariant correctly verifies the program and is good enough to prove the post-condition.

Also, I verified my program in dafny. This can be found in – '[a3/wlang/Extras/xSq\\_ySq.dfy](#)'. And it verified correctly.