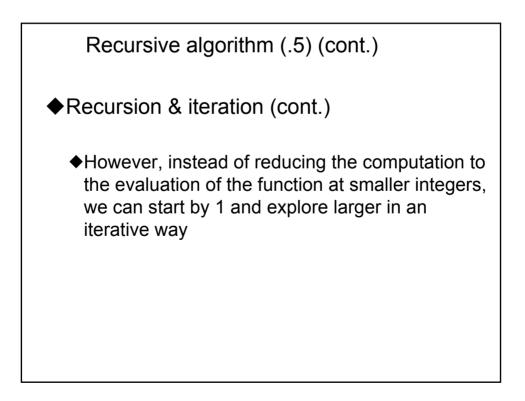


```
A recursive algorithm (.5) (cont.)

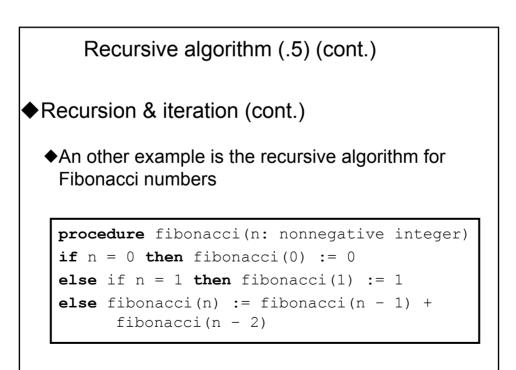
A recursive procedure for factorials

procedure factorial(n: positive integer
if n = 1 then
    factorial(n) := 1
else
    factorial(n) := n * factorial(n - 1)
```



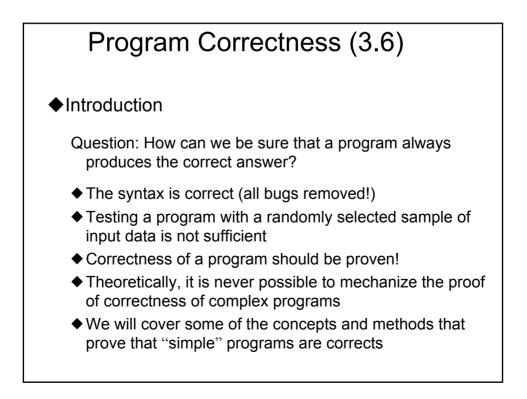
```
Recursive algorithm (.5) (cont.)
```

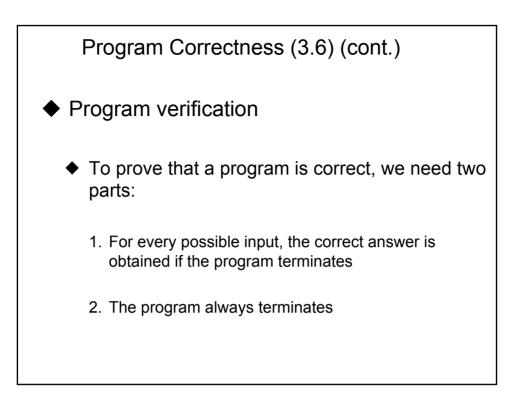
◆An iterative procedure for factorials

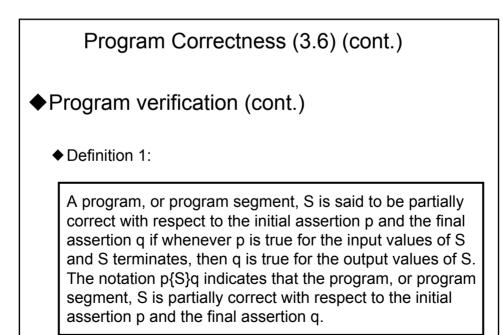


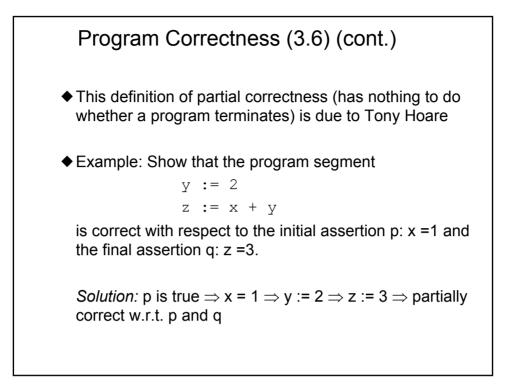
Recursive algorithm (.5) (cont.)

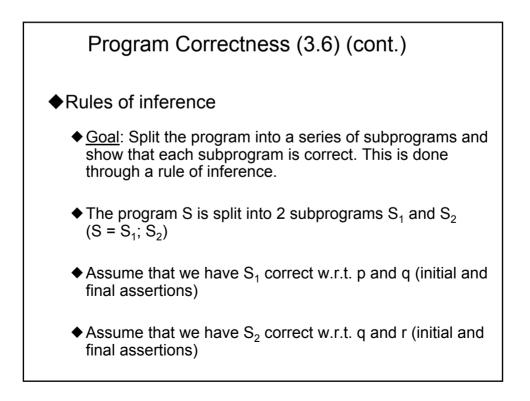
An iterative algorithm for computing Fibonacci numbers

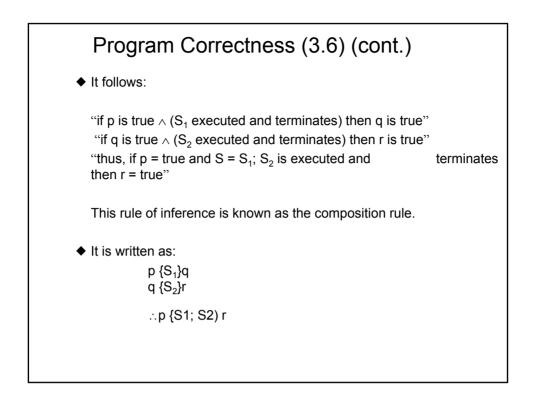


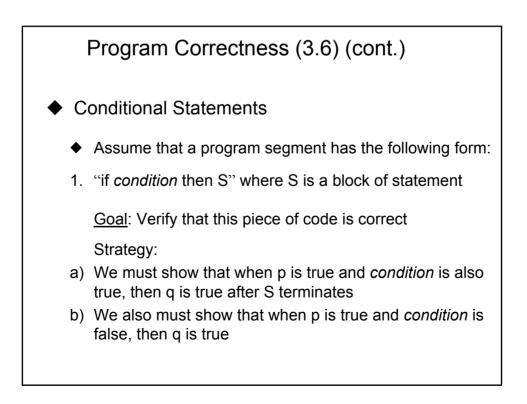


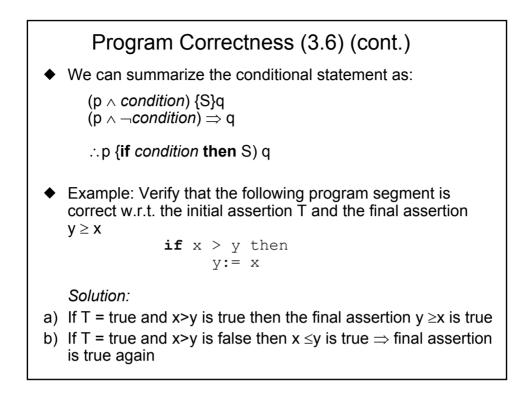


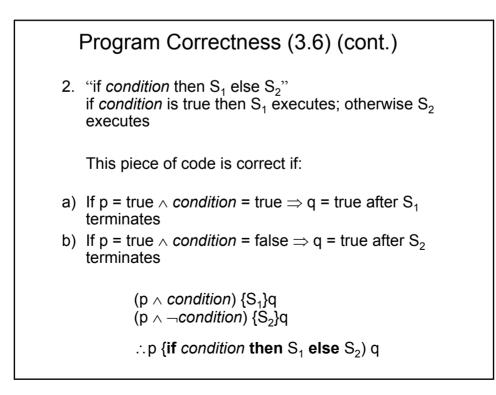


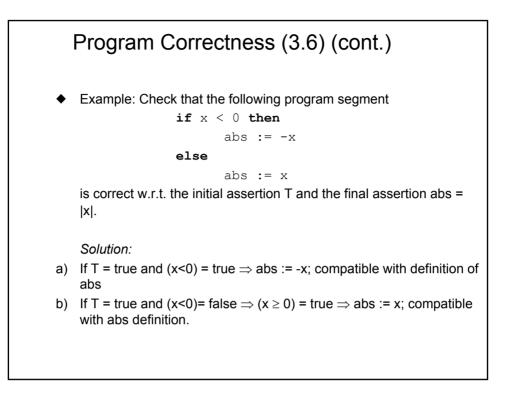


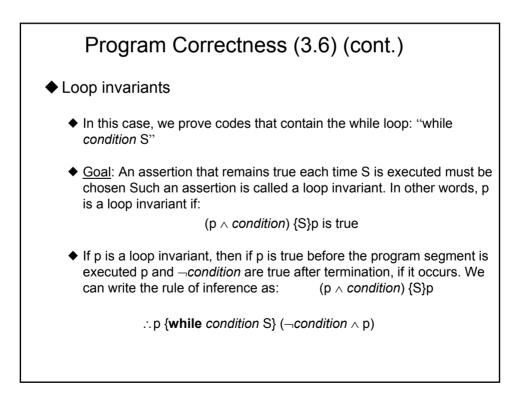


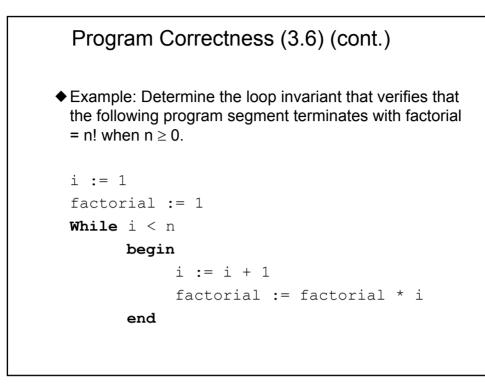


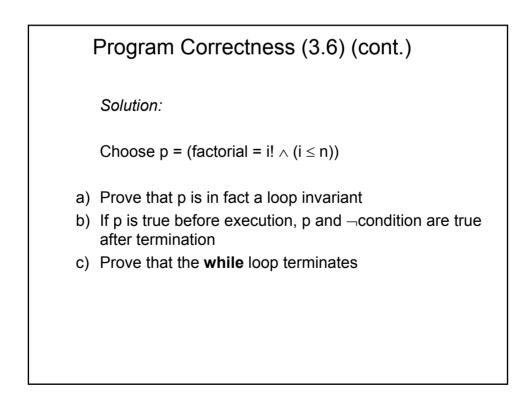












Program Correctness (3.6) (cont.)

a) P is a loop invariant:

Suppose p is true at the beginning of the execution of the **while** loop and the **while** *condition* holds;

```
\Leftrightarrow \text{factorial} = i! \land i < n
i_{new} = i + 1
\text{factorial}_{new} = \text{factorial} * (i + 1) = (i + 1)! = i_{new}!
Since i < n \Rightarrow i_{new} = i + 1 \le n
\Rightarrow p \text{ true at the end of the execution of the loop}
\Rightarrow p \text{ is a loop invariant}
```

