# Factorial Proof

- We want to prove that this recursive program computes the factorial of $n$

```
/* Factorial function definition */
int fact(int n)
{
  /* pre-condition */
  assert (n >= 1);

  /* post-condition */
  if(n > 1)
    return n * fact(n - 1);
  else
    return 1;
}
```

# Factorial Proof

- First, we recap on the exact definition of a factorial ($F!$):

$$1! = 1$$
$$2! = 1 \cdot 2 = 2$$
$$3! = 1 \cdot 2 \cdot 3 = 6$$
$$4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$$
$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$
$$\ldots$$
$$F! = F \cdot (F - 1)!$$

```
/* Factorial function definition */
int fact(int n)
{
  /* pre-condition */
  assert (n >= 1);

  /* post-condition */
  if(n > 1)
    return n * fact(n - 1);
  else
    return 1;
}
```

# Factorial Proof

- In the case of the Factorial function the integer range is $n \geq 1$, so $n = 1$ is the smallest case, and therefore would make a good base case.

```
/* Factorial function definition */
int fact(int n)
{
  /* pre-condition */
  assert (n >= 1);

  /* post-condition */
  if(n > 1)
    return n * fact(n - 1);
  else
    return 1;
}
```

**Base case**

# Factorial Proof

- **Base case**
  - fact(1) = 1! = 1

  … which we verify is correct according to the given definition of factorial

```
/* Factorial function definition */
int fact(int n)
{
  /* pre-condition */
  assert (n >= 1);

  /* post-condition */
  if(n > 1)
    return n * fact(n - 1);
  else
    return 1;
}
```

# Factorial Proof

- Next, we need to do the **inductive step**, i.e., consider the recursive case where $n > 1$
- We look carefully at the program – suppose we give some integer $k$ as input, $k > 2$, then:
  fact($k$) = $k \cdot$ fact($k - 1$)

- That is all we have to go with – if we assume that fact($k - 1$) gives the **incorrect** result then we can not say anything about fact($k$).
- So, we assume, for now, that fact($k - 1$) gives the **correct** result.

```
/* Factorial function definition */
int fact(int n)
{
  /* pre-condition */
  assert (n >= 1);

  /* post-condition */
  if(n > 1)
    return n * fact(n - 1);
  else
    return 1;
}
```

**Recursive case**

# Factorial Proof

- **Inductive step**
  - Inductive hypothesis:
    Suppose case that fact($k-1$) correctly computes the factorial of ($k-1$) for $k > 1$
  - The inductive hypothesis that fact($k-1$) is correct uses a smaller integer than fact($k$)
  - Thus, to prove the case for $k$, we need to prove the case for a case smaller than $k$

```
/* Factorial function definition */
int fact(int n)
{
  /* pre-condition */
  assert (n >= 1);

  /* post-condition */
  if(n > 1)
    return n * fact(n - 1);
  else
    return 1;
}
```

# Factorial Proof

- For any positive integer we choose, a finite chain of IF-THEN claims is created, e.g.:
  - IF fact(7) is correct THEN fact(8) is correct
  - IF fact(6) is correct THEN fact(7) is correct
  - IF fact(5) is correct THEN fact(6) is correct
  - …

```
/* Factorial function definition */
int fact(int n)
{
  /* pre-condition */
  assert (n >= 1);

  /* post-condition */
  if(n > 1)
    return n * fact(n - 1);
  else
    return 1;
}
```

# Factorial Proof

- Since we have a base case (that we've already proven) then eventually the chain of IF-THEN claims will lead to:
  - IF fact(1) is correct THEN fact(2) is correct

- We have proven that fact(1) is correct, therefore fact(2) must be correct
- … and because fact(2) is correct then fact(3) is correct
- … and so on up to fact($k$)
- Therefore fact($k$) is true for any positive integer $k$ that we choose
- … and therefore fact($n$) is true for positive integer $n$

```
/* Factorial function definition */
int fact(int n)
{
  /* pre-condition */
  assert (n >= 1);

  /* post-condition */
  if(n > 1)
    return n * fact(n - 1);
  else
    return 1;
}
```

# Factorial Proof

Therefore, the Factorial function calculates the factorial of $n$, for all integers $n \geq 1$