# Program Correctness using Induction

## Contents

Loops in an algorithm/program can be proven correct using mathematical induction. In general it involves something called "loop invariant" and it is very difficult to prove the correctness of a loop. Here we are going to give a few examples to convey the basic idea of correctness proof of loop algorithms.

**First consider the following piece of code that computes the square of a natural number:**

(We do not compute the square this way but this is just to illustrate the concept of loop invariant and its proof by induction.)

**SQUARE Function: SQ(n)**

```
S =0
i = 0
while i < n
   S = S + n
   i = i + 1
return S
```

To prove that the algorithm is correct, let us first note that the algorithm stops after a finite number of steps. For i increases one by one from 0 and n is a natural number. Thus i eventually becomes equal to n.

## loop invariant

After going through the loop k times, S = k*n and i = k hold. This statement is called a loop invariant and mathematical induction can be used to prove it.

Proof by induction.
**Basis Step**: k = 0. When k = 0, that is when the loop is not entered, S = 0 and i = 0.
Hence S = k*n and i = k hold.
**Induction Hypothesis**: For an arbitrary value m of k, S = m * n and i = m hold after going through the loop m times.
Inductive Step: When the loop is entered (m + 1)-st time, S = m*n and i = m at the beginning of the loop. Inside the loop,

S <- m*n + n
i <- i + 1

producing S = (m + 1)*n and i = m + 1.

Thus S = k*n and i = k hold for any natural number k.

Now, when the algorithm stops, i = n. Hence the loop will have been entered n times. Thus S = n*n = $n^2$. Hence the algorithm is correct.

The next example is an algorithm to compute the factorial of a positive integer.

**FACTORIAL Function: FAC(n)**

```
i = 1
F = 1
while i <= n
   F = F * i
   i = i + 1
return F
```

To prove that the algorithm is correct, let us first note that the algorithm stops after a finite number of steps. For i increases one by one from 1 and n is a positive integer. Thus i eventually becomes equal to n.

**Loop Invariant:**

After going through the loop k times, F = k ! and i = k + 1 hold. This is a loop invariant and again we are going to use mathematical induction to prove it.

Proof by induction.

**Basis Step**: k = 1. When k = 1, that is when the loop is entered the first time, F = 1 * 1 = 1 and i = 1 + 1 = 2. Since 1! = 1, F = k! and i = k + 1 hold.

**Induction Hypothesis:** For an arbitrary value m of k, F = m! and i = m + 1 hold after going through the loop m times.

*Inductive Step:* When the loop is entered (m + 1)-st time, F = m! and i = (m+1) at the beginning of the loop. Inside the loop,

F <- m!* (m + 1)

i <- (m + 1) + 1

producing F = (m + 1)! and i = (m + 1) + 1.

Thus F = k! and i = k + 1 hold for any positive integer k.

Now, when the algorithm stops, i = n + 1. Hence the loop will have been entered n times. Thus F = n! is returned. Hence the algorithm is correct.