**Base case:**

Let’s say X = 1, Y = 0.

*add(X,0)* will return the X value.

Let’s say LHS = X + Y, RHS = *add(X, Y)*

What if X = 1, Y = 1?

LHS = 1 + 1 = 2

RHS = *1 + add(X, Y-1)*

Where *add(X, Y-1) = X = 1.*

So it returns 1 + 1 = 2.

LHS = RHS.

As with each recursive call Y is decreased by 1, we will eventually reach the base case, therefore the algorithm terminates.

**Induction Step:**

Induction hypothesis: assume the algorithm is correct for a value Y = K > 0 that returns K + X.

Will the algorithm then work for K + 1?

If Y = K + 1, line 5 of the code is:

*1 + add(X, (K+1) – 1)*

Where *add(X, (K+1) – 1)* = *add(X, K).* *1 + add(X, K)* is the definition of *add(X, K+1).* This means that the algorithm will return *add(X, K+1).*

Therefore, by induction, *add(X, Y)* returns X + Y for all Y >= 0.

Simple Logic w base case

Y = K > 0

public static int add(int x, int y){

if (y == 0)

return x;

else

return 1 + add(x,y−1);

**Base case:**

LHS: simple addition; X + Y

RHS: 1 + add(x,y−1);

Y = 0

X = 2

LHS: 2 + 0 = 2

RHS: X = 2

LHS = RHS therefore the algorithm is true for the base case!

**Induction Step:**

Termination: PROVED.

Lets say K = 4

Y = 4

X =5

If Y = K + 1, line 5 of the code is:

*1 + add(X, (K+1) – 1)*

Where *add(X, (K+1) – 1)* = *add(X, K).* *1 + add(X, K)* is the definition of *add(X, K+1).* This means that the algorithm will return *add(X, K+1).*

Therefore, by induction, *add(X, Y)* returns X + Y for all Y >= 0.