The correctness for most recursive algorithms can be proved by induction. A recursive algorithm is usually made of a base condition (the point after which the program doesn’t recurse anymore), and a recursive call (which calls the same function to calculate the answer for a smaller value of the argument). You can see how this naturally draws a parallel between recursion and induction. Here are a few examples:

**Recursive algorithm for factorial** -

```python
def FactorialRecursive(n):
    if n == 0:  # base step
        return 1
    else:  # inductive/recursive step
        return n * FactorialRecursive(n - 1)
```

**Proof of correctness:**

- Base step: Ensure that the base step is correct. When \( n = 0 \), the function trivially returns 1. \( 0! = 1 \) is correct.

- Inductive step: Let’s assume that the function correctly computes the factorial for all values till \( n - 1 \), where \( n > 0 \) (induction hypothesis). Then, let’s see what happens for \( n \). As \( n > 0 \), we go straight to the else condition:

\[
    n \times \text{FactorialRecursive}(n - 1)
\]

FactorialRecursive(n-1) correctly computes the value of \((n - 1)!\) by the inductive hypothesis. Thus, this step would return \( n \times (n - 1)! = n! \). Thus, it also calculates the value of \( n! \) correctly.

From the above inductive argument, we can safely say that the given function correctly computes the value of \( n! \) for all values of \( n \).

**Recursive algorithm for raising to a power** -

```python
def PowerRecursive(base, power):
    if power == 0:  # base step
        return
    else:  # inductive/recursive step
        if power % 2 == 0:
            return PowerRecursive(base, power / 2) ^ 2
        else:
            return base * PowerRecursive(base, power / 2) ^ 2
```

**Proof of correctness:**

- Base step: Ensure that the base step is correct. When \( \text{power} = 0 \), the function trivially returns 1. \( \text{base}^0 = 1 \) for all values of \( \text{base} \) is correct.
• Inductive step: Let’s assume that the function correctly computes the power for all values till \( p - 1 \), where \( p > 0 \) (induction hypothesis). Then, let’s see what happens for \( p \). As \( p > 0 \), we go straight to the else condition:

  – power is even \((p = 2k)\): From the inductive hypothesis, we can say that the recursive call will correctly compute the value of \( \text{base}^{2k/2} = \text{base}^k \). Thus, \( \text{PowerRecursive(base, } k\text{)}^2 \) would give \((\text{base}^k)^2 = \text{base}^{2k} = \text{base}^{\text{power}}\), which is correct.

  – power is odd \((p = 2k + 1)\): \( p/2 = k \). From the inductive hypothesis, we can say that the recursive call will correctly compute the value of \( \text{base}^k \). Thus, \( \text{PowerRecursive(base, } k\text{)}^2 \) would give \((\text{base}^k)^2 = \text{base}^{2k} \). Thus, we’ll finally return \( \text{base} \times \text{base}^{2k} = \text{base}^{2k+1} = \text{base}^{\text{power}} \), which is correct.

From the above inductive argument, we can safely say that the given function correctly computes the value of \( \text{base}^{\text{power}} \) in all cases.