A fully documented version of the algorithm \texttt{maxInRange} — including the precondition and postcondition for the problem being solved, a bound function for the algorithms, and assertions that document a proof of the correctness of the algorithm, is as follows.

Indeed, there is too much documentation here! For example, it is not necessary to include the fact that "A is an input integer array with positive length $n$" at every point. The assertions listed here are being provided because there is a \textit{formal method} that can be applied, to prove that they hold, if they are given this completely. However, this is beyond the scope of these notes.

```java
// Precondition: An integer array \texttt{A}, with positive length \texttt{n}, and integers \texttt{high} and \texttt{low} such that $0 \leq \texttt{low} \leq \texttt{high} \leq \texttt{n} - 1$, are given as input.
// Postcondition: The largest element in the set 
// \{\texttt{A[low]}, \texttt{A[low+1]}, ... , \texttt{A[high]}\} 
// is returned as output.

integer maxInRange (integer[] \texttt{A}, integer \texttt{low}, integer \texttt{high}) {
    // Bound Function: \texttt{high} − \texttt{low}
    // Assertion:
    // 1. \texttt{A} is an input integer array with positive length \texttt{n}.
    // 2. \texttt{low} and \texttt{high} are integer inputs such that $0 \leq \texttt{low} \leq \texttt{high} \leq \texttt{n} - 1$.

    1. if (\texttt{low} == \texttt{high}) {
        // Assertion:
        // 1. \texttt{A} is an input integer array with positive length \texttt{n}.
        // 2. \texttt{low} and \texttt{high} are integer inputs such that $0 \leq \texttt{low} = \texttt{high} \leq \texttt{n} - 1$.
        return \texttt{A[low]}
        // Assertion:
        // 1. \texttt{A} is an input integer array with positive length \texttt{n}.
        // 2. \texttt{low} and \texttt{high} are integer inputs such that $0 \leq \texttt{low} = \texttt{high} \leq \texttt{n} - 1$.
        // 3. The value \texttt{A[low]} = \texttt{max(A[low], A[low+1], ..., A[high])} has been returned as output.
    } else {
```

```
// Assertion:
// 1. A is an input integer array with positive length n.
// 2. low and high are integer inputs such that 0 ≤ low < high ≤ n − 1.
3. integer mid := floor((low + high)/2)
   // Assertion:
   // 1. A is an input integer array with positive length n.
   // 2. low and high are integer inputs such that 0 ≤ low < high ≤ n − 1.
   // 3. mid is an integer variable such that low ≤ mid ≤ high − 1.
4. return max(maxInRange(A, low, mid), maxInRange(A, mid+1, high))
   // Assertion:
   // 1. A is an input integer array with positive length n.
   // 2. low and high are integer inputs such that 0 ≤ low < high ≤ n − 1.
   // 3. The value max(A[low], A[low+1], ..., A[high]) has been returned as output.
}
// Assertion:
// 1. A is an input integer array with positive length n.
// 2. low and high are integer inputs such that 0 ≤ low ≤ high ≤ n − 1.
// 3. The value max(A[low], A[low+1], ..., A[high]) has been returned as output.
}