1. If the index of the greatest element of A is i, it returns $(i, i, A[i])$.

2. By Strassen's algorithm, we first compute 10 S-matrices (= scalars in this particular case) by using addition and subtraction only:

$$\begin{aligned}
S1 &= B12 - B22 = 8 - 2 = 6 \\
S2 &= A11 + A12 = 1 + 3 = 4 \\
S3 &= A21 + A22 = 7 + 5 = 12 \\
S4 &= B21 - B11 = 4 - 6 = -2 \\
S5 &= A11 + A22 = 1 + 5 = 6 \\
S6 &= B11 + B22 = 6 + 2 = 8 \\
S7 &= A12 - A22 = 3 - 5 = -2 \\
S8 &= B21 + B22 = 4 + 2 = 6 \\
S9 &= A11 - A21 = 1 - 7 = -6 \\
S10 &= B11 + B12 = 6 + 8 = 14
\end{aligned}$$

The second step is to compute 7 P-matrices (= scalars in this particular case) using the multiplication only (in general this is the recursive part of the algorithm):

$$\begin{aligned}
P1 &= A11 \cdot S1 = 1 \times 6 = 6 \\
P2 &= S2 \cdot B22 = 4 \times 2 = 8 \\
P3 &= S3 \cdot B11 = 12 \times 6 = 72 \\
P4 &= A22 \cdot S4 = 5 \times (-2) = -10 \\
P5 &= S5 \cdot S6 = 6 \times 8 = 48 \\
P6 &= S7 \cdot S8 = (-2) \times 6 = -12 \\
P7 &= S9 \cdot S10 = (-6) \times 14 = -84
\end{aligned}$$

The final step is to form the product $C = AB$, which uses addition and subtraction only:

$$\begin{aligned}
C11 &= P5 + P4 - P2 + P6 = 48 - 10 - 8 - 12 = 18 \\
C12 &= P1 + P2 = 6 + 8 = 14 \\
C21 &= P3 + P4 = 72 - 10 = 62 \\
C22 &= P5 + P1 - P3 - P7 = 48 + 6 - 72 + 84 = 66
\end{aligned}$$

3. ```
Strassen(A,B)
   n = size(A)
   create a new n by n matrix C
   if n == 1
      C(1,1) = A(1,1)*B(1,1)
   else
      partition A into 2 by 2 block matrices denoted as A11, A12, A21, A22
      partition B into 2 by 2 block matrices, denoted as B11, B12, B21, B22
      // Compute 10 S-matrices by addition and subtraction only
      S1 = B12 - B22
      S2 = A11 + A12
      S3 = A21 + A22
      S4 = B21 - B11
      S5 = A11 + A22
      S6 = B11 + B22
      S7 = A12 - A22
      S8 = B21 + B22
      S9 = A11 - A21
      S10 = B11 + B12
      // Compute 7 P-matrices by multiplication, recursively
      P1 = Strassen(A11, S1)
      P2 = Strassen(S2, B22)
      P3 = Strassen(S3, B11)
      P4 = Strassen(A22, S4)
      P5 = Strassen(S5,S6)
      P6 = Strassen(S7,S8)
      P7 = Strassen(S9,S10)
      // Compute blocks of the product C by addition and subtraction only
      C11 = P5 + P4 - P2 + P6
      C12 = P1 + P2
      C21 = P3 + P4
      C22 = P5 + P1 - P3 - P7
      combine C11, C12, C21 and C22 into C
   end if
   return C
```

4. Use the divide-and-conquer integer multiplication algorithm to multiply the two binary integers 10011011 and 10111010.

   Write

$$
\begin{aligned}
x &= 10011011 = 1001 \cdot 2^4 + 1011 \equiv x_L \cdot 2^4 + x_R \\
y &= 10111010 = 1011 \cdot 2^4 + 1010 \equiv y_L \cdot 2^4 + y_R
\end{aligned}
$$

   Then

$$
\begin{aligned}
x_L + x_R &= 1001 + 1011 = 10100 \\
y_L + y_R &= 1011 + 1010 = 10101 \\
(x_L + x_R)(y_L + y_R) &= 10100 \times 10101 = 110100100 \\
x_L y_L &= 1001 \times 1011 = 1100011 \\
x_R y_R &= 1011 \times 1010 = 1101110 \\
x_L y_R + x_R y_L &= (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R \\
&= 110100100 - 1100011 - 1101110 = 11010011
\end{aligned}
$$
```

Therefore

$$\begin{aligned} xy &= 2^8 x_L y_L + 2^4(x_L y_R + x_R y_L) + x_R y_R \\ &= 2^8 \cdot 1100011 + 2^4 \cdot 11010011 + 1101110 \\ &= 111000010011110 \end{aligned}$$

Note that in a full implementation of the divide-and-conquer integer multiplication algorithm, the products $(x_L + x_R)(y_L + y_R)$, $x_L y_L$ and $x_R y_R$ are computed recursively, using the divide-and-conquer. However, for the simplicity, they are computed directly.

5. Each function call prints one line and calls the same function on input of half the size, so the number of printed lines is

$$P(n) = 2P(n/2) + 1.$$

By iteration, note that $n = 2^k$, we have

$$\begin{aligned} P(n) &= 2P(n/2) + 1 \\ &= 2(2P(n/2^2) + 1) + 1 = 2^2 P(n/2^2) + 2 + 1 \\ &= 2^2(2P(n/2^3) + 1) + 2 + 1 = 2^3 P(n/2^3) + 2^2 + 2 + 1 \\ &= \cdots \\ &= 2^k P(n/2^k) + 2^{k-1} + \cdots + 2 + 1 \\ &= 2^{k-1} + \cdots + 2 + 1 \quad \text{(note: } P(1) = 0\text{)} \\ &= 2^k - 1 = n - 1 \end{aligned}$$

6. (a) *Algorithm idea:* Suppose we want to search S between indices ℓ and r, initially $\ell = 1$ and $r = n$. If $\ell > r$, we are done and answer that there is no index i such that $S[i] = i$. Else look at $S[m]$, where $m = \lfloor (\ell + r)/2 \rfloor$. If $S[m] = m$, again we're done. Otherwise if $S[m] > m$, recursively search S between ℓ and $m - 1$ (the left half), while if $S[m] < m$, recursively search A between $m + 1$ and r (the right half).

(b) *Correctness:* Why does this work? Because we are dealing with distinct and sorted integers $S[i]$. When we learn that $S[m] > m$, we know that $S[m+1] > m+1$, and so forth, and therefore, we only need to continue the search in the left half of the array, i.e., $S[\ell ... m - 1]$. Similarly for the case of $S[m] < m$.

(c) *Pseudocode:*

```
FindEqIndex(S)
   FindEqIndexRec(S,1,length(S))
end

FindEqIndexRec(S,low,high)
if (low > high) then
   print (``No solution'')
else
   mid = floor((low+high)/2)
   if S[mid] == mid
      print(``S[mid] = mid'')
   else
      if S[mid] < mid then
         FindEqIndexRec(S,mid + 1,high)
      else // S[mid] > mid
         FindEqIndexRec(S,low,mid - 1)
```

```
      end if
    end if
  end if
```

(d) *Running time:*

$$T(n) = T(n/2) + 1.$$

By iteration, assume $n = 2^k$, the solution of the recursion is

$$T(n) = T(n/2) + 1. = T(n/2^2) + 2 = \cdots = T(n/2^k) + k = T(1) + k = 1 + \lg n$$

Or by the master theorem, we have $T(n) = \Theta(\lg n)$.

7. (a) Let $T(i)$ be the time to merge arrays 1 to i. This consists of the time taken to merge arrays 1 to $i-1$ and the time taken to merge the resulting array of size $(i-1)n$ with array i, i.e. $O(in)$. Hence, for some constant c,

$$T(i) = T(i-1) + cni.$$

By solving the above recursion, we have

$$T(k) = T(1) + cn\sum_{i=2}^{k} i = O(nk^2).$$

(b) Divide the arrays into two sets, each of $k/2$ arrays. Recursively merge the arrays within the two sets and finally merge the resulting two sorted arrays into the output array. The base case of the recursion is $k = 1$, when no merging needs to take place. The running time is given by

$$T(k) = 2T(k/2) + O(nk).$$

By the master theorem,

$$T(k) = O(nk \log k).$$