# UG1-2

**a)** Given two bit arrays $A[1 \ldots n-1]$ and $B[1 \ldots n-1]$ where each array represents an n-bit number, return the array representing the largest value.
**b)**

```
1: procedure MaxInt(arrayA, arrayB)
2:     for k ← n − 1 to 0 do
3:         if A[k] ≠ B[k] then
4:             if A[k] = 1 then
5:                 return A
6:             else
7:                 return B
8:     return A
```

**c)**
**Loop Invariant:** At the start of each iteration of the for loop all bits $> k$ in array $A$ and $B$ are equal.

**Initialization:** In the first iteration the loop invariant holds since there are no bits above $k$.

**Maintenance:** Given two binary values, 10101 and 10110, by beginning at the most significant bit and scanning until the bits are no longer identical the larger value can be identified by selecting the number with the 1 in the current digit. This is because the value corresponding to the shared digit can be subtracted from each without affecting the ability to determine the largest number. The same is true for decimal digits. For example, 1234 and 1245. Subtracting 1200 with 34 and 45 still allows us to determing the maximum integer.

In each iteration, at line 3, if $A[k] \neq B[k]$ the code tests the bit stored in $A[k]$. If this bit is a 1 then the value represented by array $A$ must be larger since the $B[k]$ bit must be a 0 so the solution $A$ is returned. Otherwise, if $A[k] = 0$ then $B$ must be returned as the solution since $B[k] = 1$ and thus is the larger value.

If $A[k] = B[k]$ then the loop continues and the loop invariant holds.
**Termination:** The loop will terminate when 1 of 2 conditions are met,

1. If $A[k] \neq B[k]$ the procedure will return the array representing the largest value.
2. If MaxInt reaches line 8 then all $n$ bits of $A$ and $B$ were equal so just return $A$.

d) Since the worst case is if all $n$ bits are equal the running time is $O(n)$