# Q1-

**a)**

```
MergeSort(A)                          //O(n ln(n))
Return(A[ceil(size(A) / 2)])          //O(1)
```

**b)**

```
FindMajority(A)
{
        If size(A) == 1
                Return A[1]

        L = FindMajority(A[1...ceil(size(A) / 2)])
        R = FindMajority(A[ceil(size(A) / 2)+1…size(A)])

        Lc = Rc = 0;
        For i = 1 to size(A)
                Lc += (A[i] == L)
                Rc += (A[i] == R)
        End

        If  Lc > Rc  Then  Return L  Else  Return R
}

// T(n) = 2 T(n / 2) + O(n) which is in O(n ln(n))
```

**c)**

```
X = A[1];
Cnt = 1;
For i = 2 to size(A)                  //O(n)
        If  Cnt == 0  Then  X = A[i]
        If  A[i] == X  Then  Cnt++  Else Cnt--
End
Return X
```

# Q2-

**a)**

False; by Counter Example: g(n) = ln(n) and f(n) = ln(n^2) = 2 ln(n)
==> e^g(n) = n and f^g(n) = n^2 and we know that n^2 is not in O(n)

**b)**

False or True based on your assumptions!
FALSE - By Counter Example:
f(n) = e^(1/n) and g(n) = e^(1+1/n) = e * e^(1/n)
But ln(f(n)) = 1/n is not in ln(g(n)) = 1 + 1/n

TRUE - if g(n) is an increasing function AND there exists m such that g(m) > 1:
We know f(n) is in O(g(n)). So for some c and n0 we have
for all n > n0: f(n) <= c g(n) ==> ln(f(n)) <= ln(c g(n)) = ln(c) + ln(g(n))
Since g(n) is increasing, if n1 = max(n0, m) then for all n > n1: g(n) > 1
Set k = 1 + ln(c) / ln(g(n1)) so we have:
k ln(g(n)) = ln(g(n)) + ln(c) ln(g(n)) / ln(g(n1)) >= ln(c) + ln(g(n)) >= ln(f(n))
==> these exists k and n1 such that for all n > n1: ln(f(n)) <= k ln(g(n))
==> So ln(f(n)) is in O(ln(g(n))

## Q3-

Only ONE class --> ln(n) and log(n)
The answer is:
10^9 < ln(ln(n) < log(n) = ln(n) < n^0.0000001 < (4/3)^log(n) < sqrt(n) < n < n^2 < n^2 ln(n) < 50n^3 < ln(n)^ln(n) < n^log(n) < n^ln(n) < 100e^sqrt(n) < (4/3)^n < 2^n < e^n < n! < (n+5)! < 2^2^n

## Q4-

There are many examples, like this one:

```
f(n) =  {  2(n-1)!       if n is odd
        {  n!            if n is even
        { 1              if n == 1

g(n) =  {  n!            if n is odd
        {  2(n-1)!       if n is even
        { 1              if n == 1
```

Obviously they are increasing functions from N to N.
Proof by contradiction:
Assume f(n) is in O(g(n)) which means there exists c and n0 such that
for all n > n0: f(n) <= c g(n)
Now take any even m > max(2c, n0), we have:
f(m) = m! and g(m) = 2(m-1)!
So we must have: m! <= c 2(m-1)!
Which means m <= 2c [Contradiction]
In the same way we can prove g(n) is not in O(f(n)) .

## Q5-

```
For i = size(A) to 2
        Tmp = A[i – f[i]]
        For j = i – f[i] to i-1
                A[j] = A[j+1]
        End
        A[i] = Tmp
End
```