Divide And Conquer

Main Focus: Calculating k^n effectively

Intent:

To calculate k^n.

Naive Algorithm:

The naive algorithm to compute k^n is: Start with 1 and multiply by k until reaching k^n. For this approach, there are $n-1$ multiplications and each takes constant time giving a $O(n)$ algorithm.

Faster approach to compute k^n.

Example: $9^{24} = (9^{12})^2 = ((9^6)^2)^2 = (((9^3)^2)^2)^2 = ((((9^2).9)^2)^2)^2$

When we take squares of numbers, we only need 5 multiplications instead of 23. the algorithm will be:

```
int Exponential(int k, int n){
    if(k==0)
        return 1;
    else{
        if(n%2==1)
            a = Exponential(k,n-1);
            return a*k;
    }
    else{
        a = Exponential(k,n/2);
        return a*a;
    }
}
```

Time Complexity: $T(n) = O(logn)$. We can get using Masters Theorem (discussed earlier).
