1. *Find the complexity of following algorithm.*

```
int powerA(int x, int n)
  {
    if (n==0)
      return 1;
    if (n==1)
      return x;
    else
      return x * powerA(x, n - 1);
  }
```

SOLUTION: IF $n = 1$*,then the result is x. Returning x takes unit time .IF* $n > 1$*, then the algorithm enters into the else part . This part contains a multipplication and a same algorithm repetition up to n-1 time .So, we can write* $T(n-1) + 2 if n > 1$.
So, we can write.

$$T(n) = T(n-1) + 2 \quad \text{if} \quad n > 1$$
$$T(n) = 1 \quad \text{if} \quad n = 1$$

$$T(n) = T(n-1) + 2$$
$$T(n-1) = T(n-2) + 2$$
$$T(n-2) = T(n-3) + 2$$

$$T(2) = T(1) + 2$$
$$T(1) = 1$$

Adding all these, we get

$$T(n) = (n-1) * 2 + 1 = 2n - 1$$

So, the complexity is O(n). Here, $2n - 1$ is a polynomial of n of degree 1.

2. Write the divide and conquer algorithm for merge sort and find its complexity.[Sri Venkateswara University 2008]
SOLUTION: In merge sort, the n element array is divided into tow halves each with n/2 elements (for n = even), or one with n/2 and the other with n/2 +1 elements. The two halves and merged. The algorithm for performing merge sort is as follows:

```
mergeSort(int min, int max)
{
  if(min < max)
    {
      m = (min + max)/2
      mergeSort(min, m)
      mergeSort(m + 1, max)
      Merge(min, m, max)
    }
}
```

```
Merge(min, m, max)
{
  i = min, j = min + 1, k = 0
  while((i ≤ min) && j ≤ max)
    {
      if(A[i] < A[j])
      {
        temp[k++] = A[i]
        i++
      }
    elseif(A[i] > A[j])
      {
        temp[k++] = A[j]
        j++
      }
        else
        {
          temp[k++] = A[i]
          temp[k++] = A[j]
        }
    }
  while(i ≤ m)
  {
    temp[k++] = A[i++]
  }
  while(j ≤ q)
  {
    temp[k++] = A[j++]
  }
  for(i = 0, i < n, i++)
  {
  A[i] = temp[i]
  }
```

The algorithm Merge (min, m, max) is performed in O(n) .The algorithm mergesort (int min, int max) of n element becomes half when it is called recursively. Thus, the whole algo-rithm takes

$$T(n) \leq 2T\left(\frac{n}{2}\right) + cn$$

$$\leq 2\left[2T\left(\frac{n}{2^n}\right) + c\frac{n}{2}\right] + cn$$

$$T(n) \leq 2^2 T\left(\frac{n}{2^2}\right) + 2cn$$

.....

.....

$$T(n) \leq 2^i T\left(\frac{n}{2^i}\right) + icn$$

$$If\, 2^i = n, i = log_2 n$$

$$T(n) \leq 2^i T(1) + cnlog_2 n$$

T(1) =1 (time required to sort a list of one element).
Thus, the complexity of merge sort is O(n $log_2 n$).