Rojin and Suresh Assignment 4

1. Pseudo-code recursive method, findMax(L),

```
function findMax(array):
   // Base case: if array is empty, return null
   if array is empty:
       return null;

   // Get the first number of array
   firstElement = array[0];

   retun maxHelper(arrat, 1 , firstElement);

function maxHelper(array, index, currentMax)
 if index = array.length
   return currentMax
 else
   currentMax = Math.max(currentMax, array[index]);
   index++;
   return maxHelper(array, index, macurrentM)
```

2. Write a pseudo code function, *sum*(n), to recursively sum the first n natural numbers but divide the problem in half and make two recursive calls. [Refer Decrease and Conquer Approach] – Example: Multiple Recursion.

```
Time complexity is O(logn)
function sum(n):
    // Base case: if n is 0, return 0
    if n == 0:
        return 0
    else if n == 1:
        return 1
    else:
        // Split the problem into two halves and make two recursive calls
        mid = Math.floor(n / 2)
        sumLeft = sum(mid)
        sumRight = sum(n - mid)

        // Combine the results of the two halves
        return sumLeft + sumRight
```

3. Pseudo code function, *isEven*(n)

I implemented this in class so I didn't want to Reimplement Mutual Recursion.

```
export function isEven(num: number): boolean {
  if (num < 0) return false;
  else if (num === 0) return true;
  else if (num === 1) return false;
  else return isEven(num - 2);
}
```

4. Recursive pseudo code function, power(x, k)

O(k)

```
export function power(x: number, k: number): number {
  if (k === 0) return 1;
  else if (k === 1) return x;
  else {
    return x * power(x, k - 1);
  }
}
```

Yes we can do this Problem in Log(k).