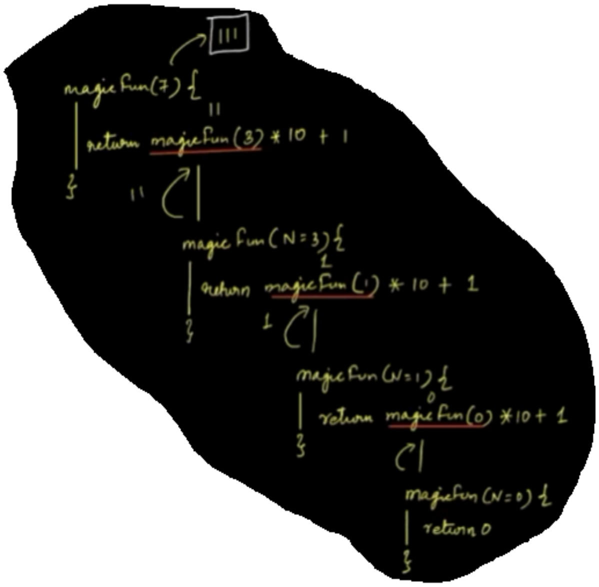
Recursion3

Quiz1

Find the output of below

```
Int magicFun(int N){
        if(N == 0){
                return 0;
        }else{
                Return magicFun(N/2)*10 +
(N%2);
        }
}
```

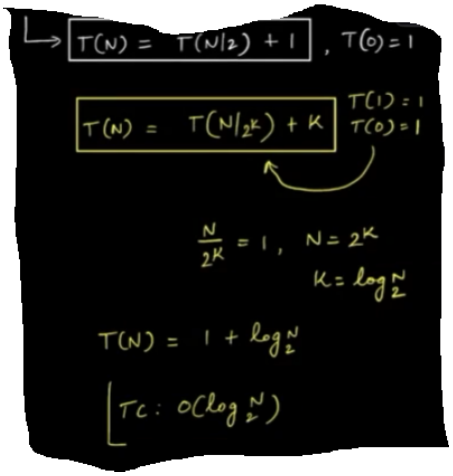
For N = 7



```
                  111
magic fun(7) {
                  11
| return magicfun(3) * 10 + 1
}
        11
    magic fun(N=3){
                       1
    | return magicfun(1) * 10 + 1
    }
              1
        magicfun(N=1){
                        0
        | return magicfun(0) *10 + 1
        }

            magicfun(N=0){
            | return 0
            }
```

T.c:



$\boxed{T(N) = T(N/2) + 1}$, $T(0)=1$

$\boxed{T(N) = T(N/2^K) + K}$ $T(1)=1$ $T(0)=1$

$\frac{N}{2^K} = 1$, $N = 2^K$

$K = \log_2^N$

$T(N) = 1 + \log_2^N$

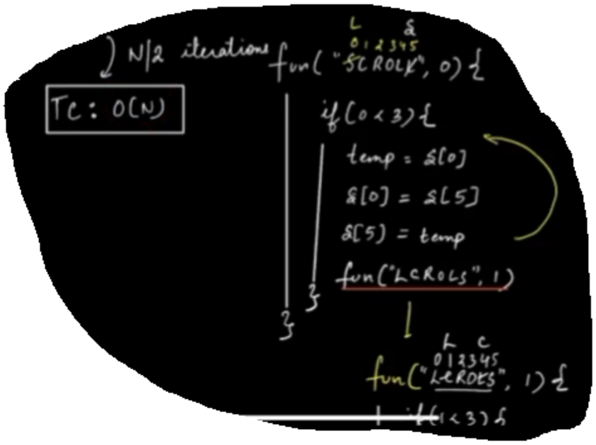
$Tc : O(\log_2^N)$

Quiz 2

Find the output for fun(“SCROLL”)

```
Void fun(char s[], int x){
        print(s);
        Char temp;
        if(x < s.len/2){
                Temp = s[x]
                S[x] = s[s.len-x-1]
                S[s.len-x-1] = temp;
                fun(s, x+1)
        }
}
```

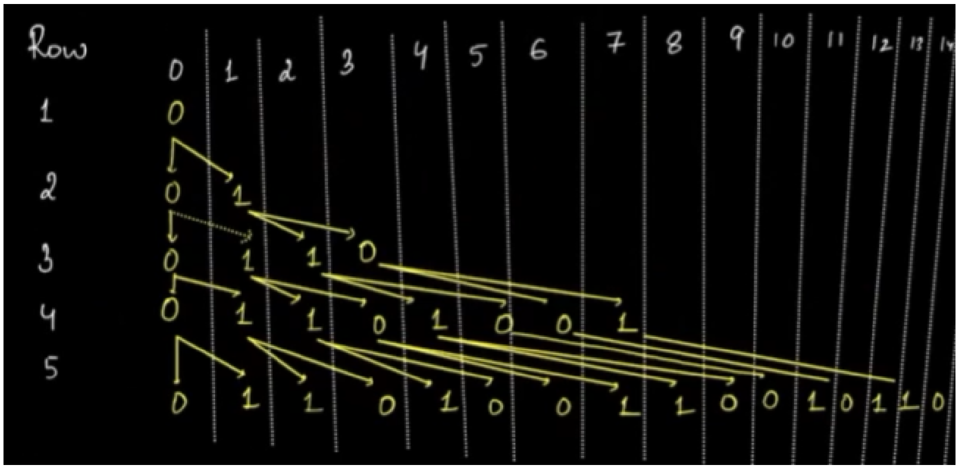
T.c = O(N)



```
} N/2 iterations      L    S
                      012345
Tc : O(N)        fun("SCROLL", 0) {
                    if(0 < 3){
                        temp = s[0]
                        s[0] = s[5]
                        s[5] = temp
                        fun("LCROLS", 1)
                    }
                 }
                                L  C
                                012345
                          fun("LCROLS", 1) {
                            if(1 < 3) {
```

Problem 1:

Kth Symbol
Each row is generated by replacing all elements of the previous row.
1 -> 1 0
0 -> 0 1

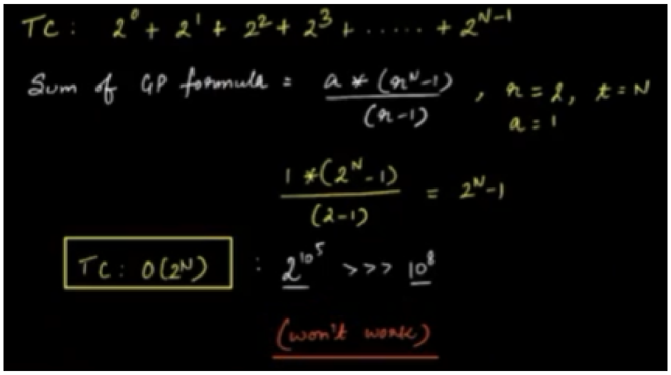


Input : N and K f{find the value at Kth index in Nth row.}
4 5 -> return 0
5 11 -> return 1
4 9 -> Not possible

Brute Force:
Construct Nth Row and find Kth Element.

Nth -> N-1th -> N-2nd -> ….. -> 3 -> 2 -> 1
2^2 2^1 0

For any ith Row = 2^(i-1)



$TC: 2^0 + 2^1 + 2^2 + 2^3 + \dots\dots + 2^{N-1}$

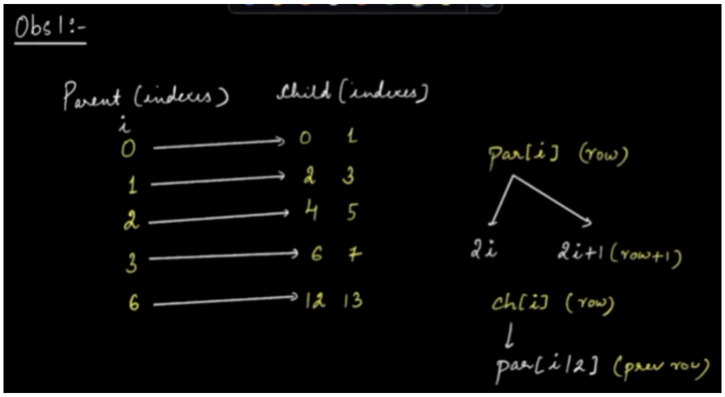
Sum of GP formula = $\frac{a*(r^N-1)}{(r-1)}$, $r=2$, $t=N$, $a=1$

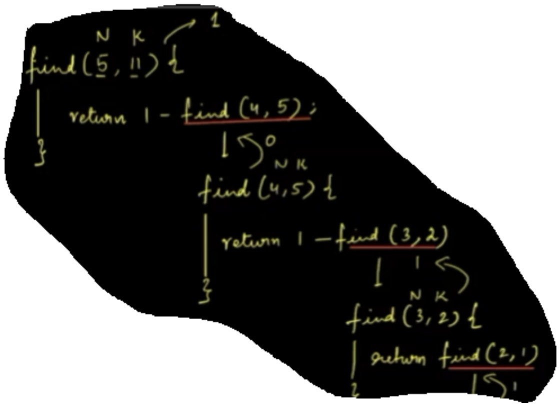
$\frac{1*(2^N-1)}{(2-1)} = 2^N-1$

$\boxed{TC: O(2^N)}$: $\underline{2^{10^5}} >>> \underline{10^8}$

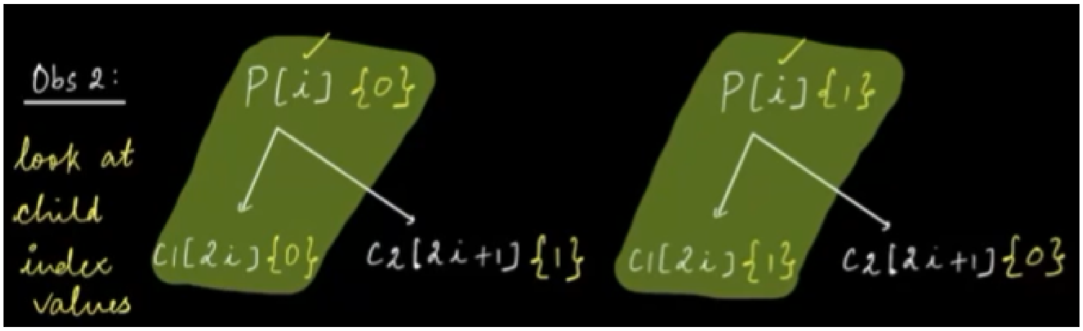
(won't work)

Optimised Approach





```
   N  K      1
find(5, 11) {
 | return 1 - find(4, 5);
}                  |  0
                       N K
              find(4,5) {
               | return 1 - find(3, 2)
              }                 |  1
                                  N K
                             find(3, 2) {
                              | return find(2, 1)
                             }          |  1
```



Note:
From above two observation:
For any child, if the index is even : value will be the same as parent.
If the child index is odd : value will be opposite of parent.

Now, To find element at: Nth Row and Kth Column

Pseudocode:

```
int find(int N, int K){
        if(k==0){
                return 0;
        }
        if(k%2 == 0){
                return find(N-1, k/2);
        }else{
                return 1 - find(N-1, k/2);  // 1 - What is returned from the parent=Opp
value
        }
}
```

T.C : T(K) = T(K/2) +1 | *T.C : O($\log_2 K$)*

Problem 2:

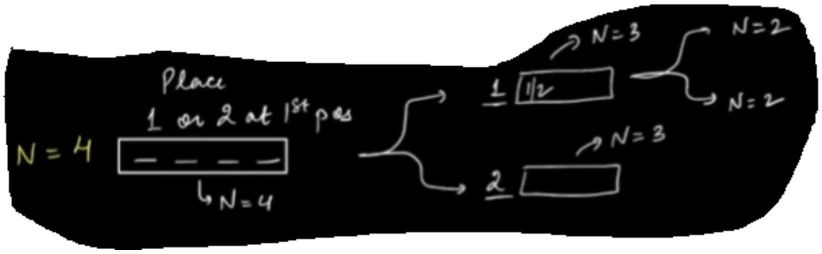
Given N digits Print all N digits formed by only 1 and 2 in increasing order of numbers.

N = 2

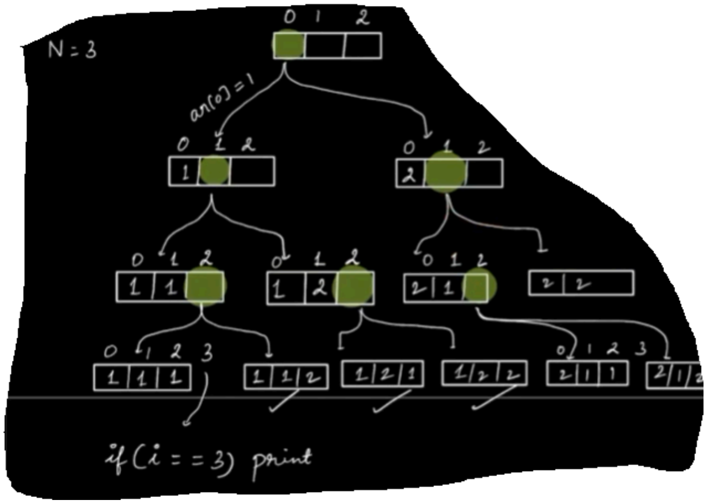
    1 1
    1 2
    2 1
    2 2

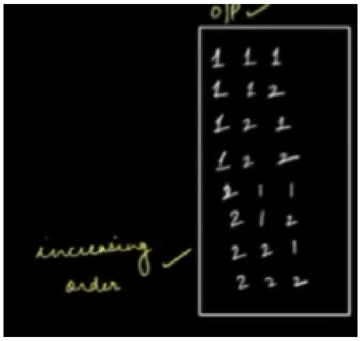
N = 3

    1 1 1
    1 1 2
    1 2 1
    1 2 2
    ……..



Idea:- Generate all N digit numbers using Recursion.





o/p ✓
1 1 1
1 1 2
1 2 1
1 2 2
2 1 1
2 1 2
2 2 1
2 2 2
increasing order ✓

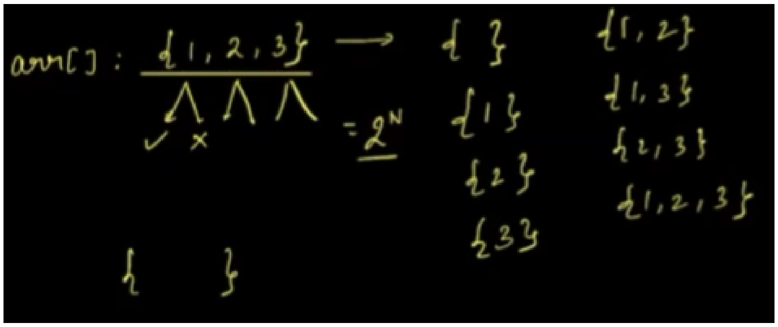
```
Void printAll (int arr[], int i, int N){
        if(i == N){
                Iterate and print array // and.push(arr);
                return;
        }

        // At ith index i've two choices
        Arr[i] = 1 // Choice 1
        printAll(arr, i+1, N);
        Arr[i] =2 // Choice 2
```

```
                printAll(arr, i+1, N);
        }
```

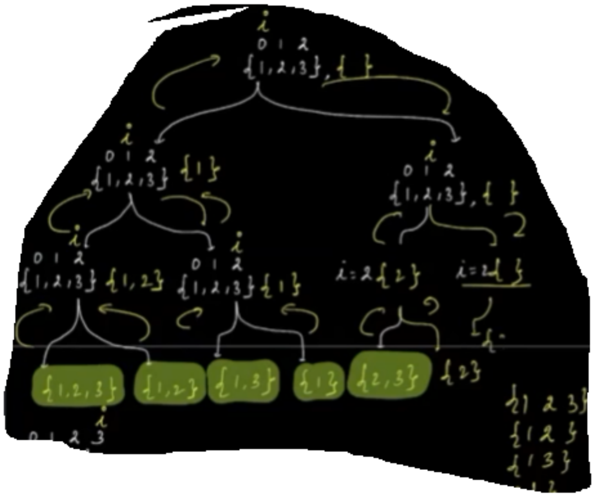
Problem 3

Given an array. Print all the subsets using recursion



Every element in the array has 2 options

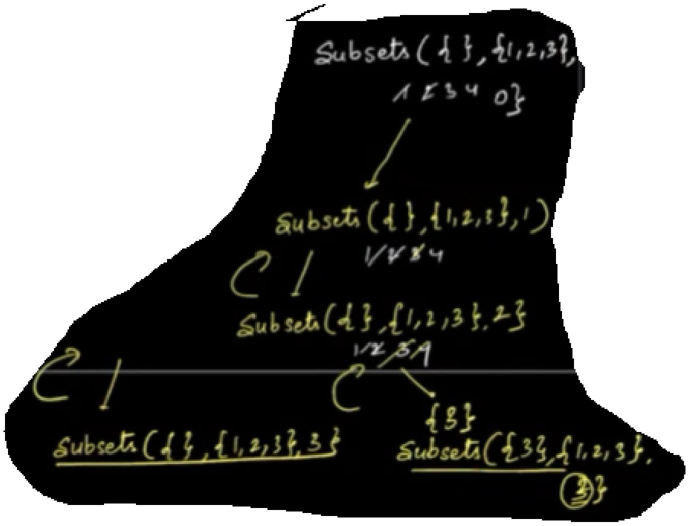
1. Considered in the subset
2. Not considered in the subset.



if(i == N){ // done forming all the subsets}

```
list<list<int> ans;
void subsets(list<int> curr, list<int> arr, int
i){
        if(i == arr.length){
                ans.add(curr);
                return;
        }

        subsets(curr, arr, i+1);
        curr.push(arr[i]);
        subsets(curr, arr, i+1);
}
```


Subsets({ }, {1,2,3}, 0}
Subsets({ }, {1,2,3}, 1)
Subsets({ }, {1,2,3}, 2}
Subsets({ }, {1,2,3}, 3}
{3}
Subsets({3}, {1,2,3}, 3}