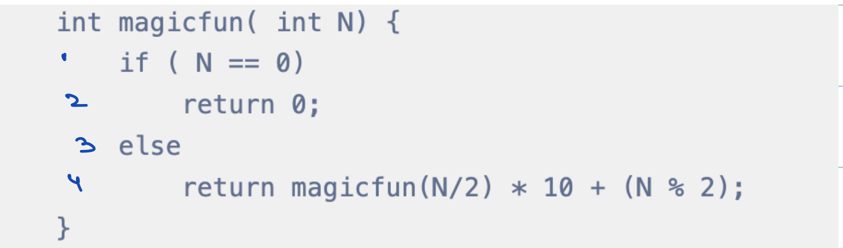
Que

N = 7

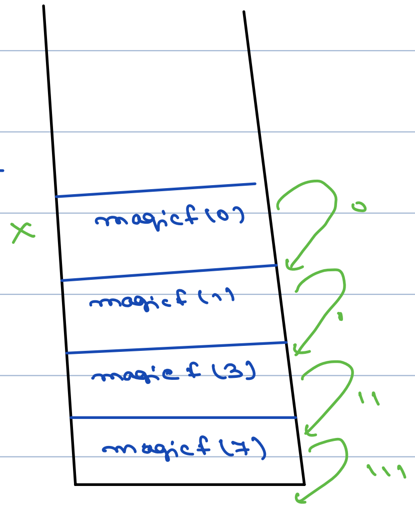


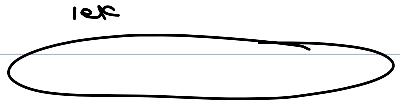
```
int magicfun( int N) {
    if ( N == 0)
        return 0;
    else
        return magicfun(N/2) * 10 + (N % 2);
}
```

T.C -> no. of fncn call × Time of fnc call

-> O(logn)

S.C -> O(logn)





10k

int [] arr = new int[]

10k

0 1 2 3 4 5

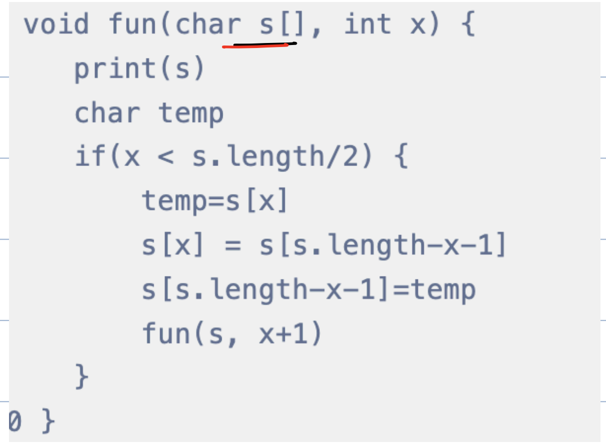
char s[] = ['~~S~~', '~~C~~', '~~R~~', '~~O~~', '~~L~~', '~~L~~'];

L L O R C S

Quese

fun(s , 0)

fun(10k, 0);



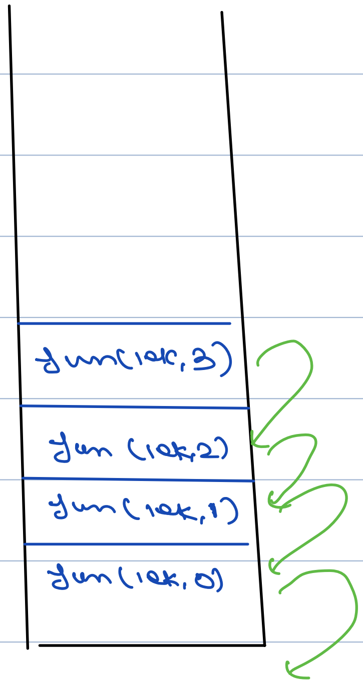
```
void fun(char s[], int x) {
   print(s)
   char temp
   if(x < s.length/2) {
       temp=s[x]
       s[x] = s[s.length-x-1]
       s[s.length-x-1]=temp
       fun(s, x+1)
   }
0 }
```

SCROLL

LCROLS

LLROCS

LLORCS



T.C → O(n)

S.C → O(n)

Ques

$arr[\,] = \{1, 2, 3\} \rightarrow 2^n$.

(Count of subsets)

Print all subsets.

$\{\}$

$\{1\}$

$\{2\}$

$\{3\}$

$\{1, 2\}$

$\{2, 3\}$

$\{1, 3\}$

$\{1, 2, 3\}$

$arr[\,] \rightarrow \{1, 2, 3, 4\} \rightarrow 2^n \rightarrow 16$.

$\{\}$

$\{1\}$

$\{2\}$

$\{3\}$

$\{1, 2\}$

$\{2, 3\}$

$\{1, 3\}$

$\{1, 2, 3\}$

$\{4\}$

$\{1, 4\}$

$\{2, 4\}$

$\{3, 4\}$

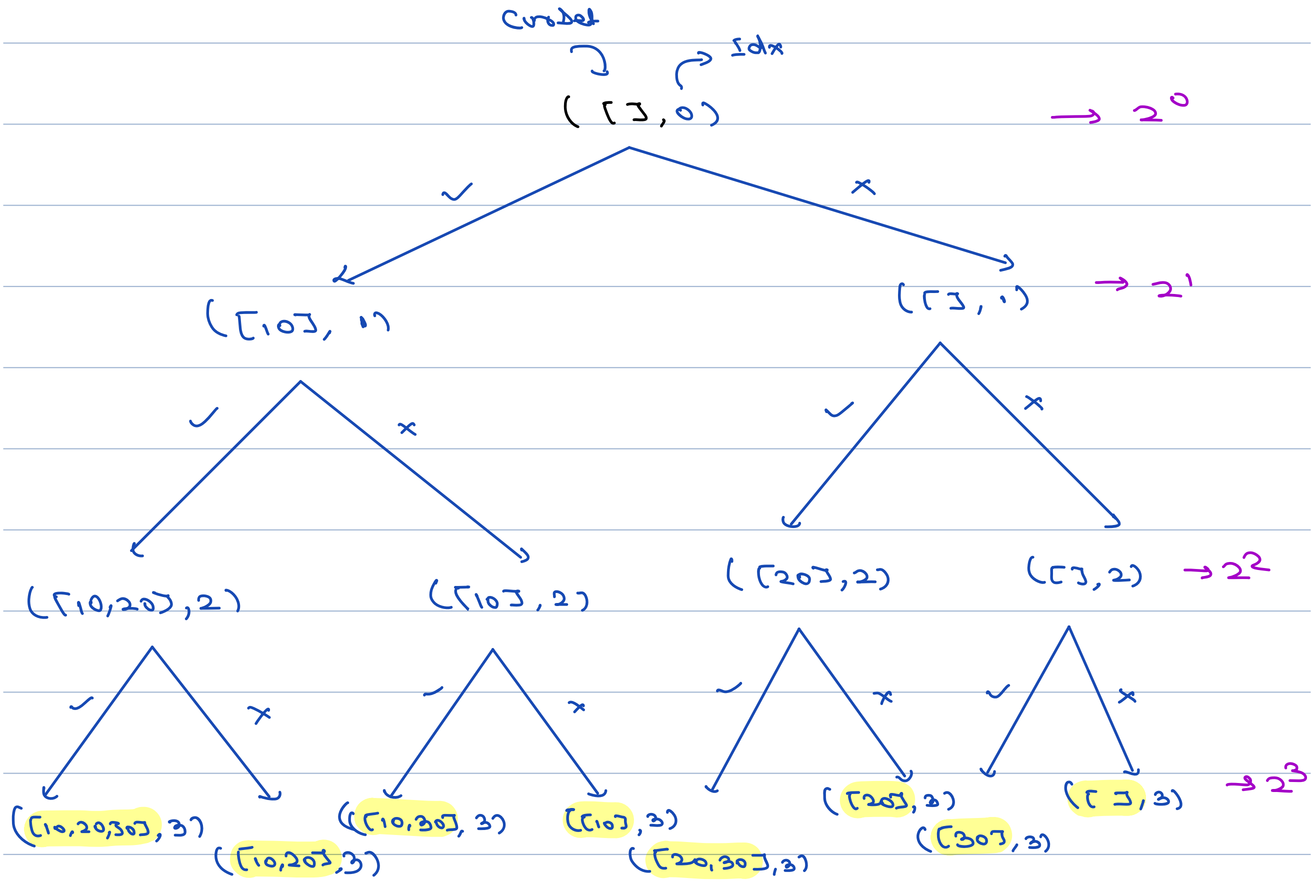
$\{1, 2, 4\}$

$\{2, 3, 4\}$

$\{1, 3, 4\}$

$\{1, 2, 3, 4\}$

$$\text{arr}[\,] \rightarrow \{\overset{0}{10}, \overset{1}{20}, \overset{2}{30}\}$$



$n = 3$

$$\underline{2^0 + 2^1 + 2^2 - 2^3}$$

$\underline{n}$

$$2^0 + 2^1 + 2^2 + 2^3 + \cdots 2^n$$

↓

2^{n+1}

T.C → $O(2^n)$

S.C → $O(\underline{n})$

```
List <List < Integer>> ans;
void subsets (int []arr , int idx , List <int> curset){
        if (idx == n) { ans.add(copy(curset)) return }

        // for every element two choices.
        1) Pick .

         curset.add(arr[idx]);
         subsets (arr, idx+1, curset);
         curset.remove(curset.size()-1);

        2) don't Pick

         subsets (arr, idx+1, curset);

    }
```

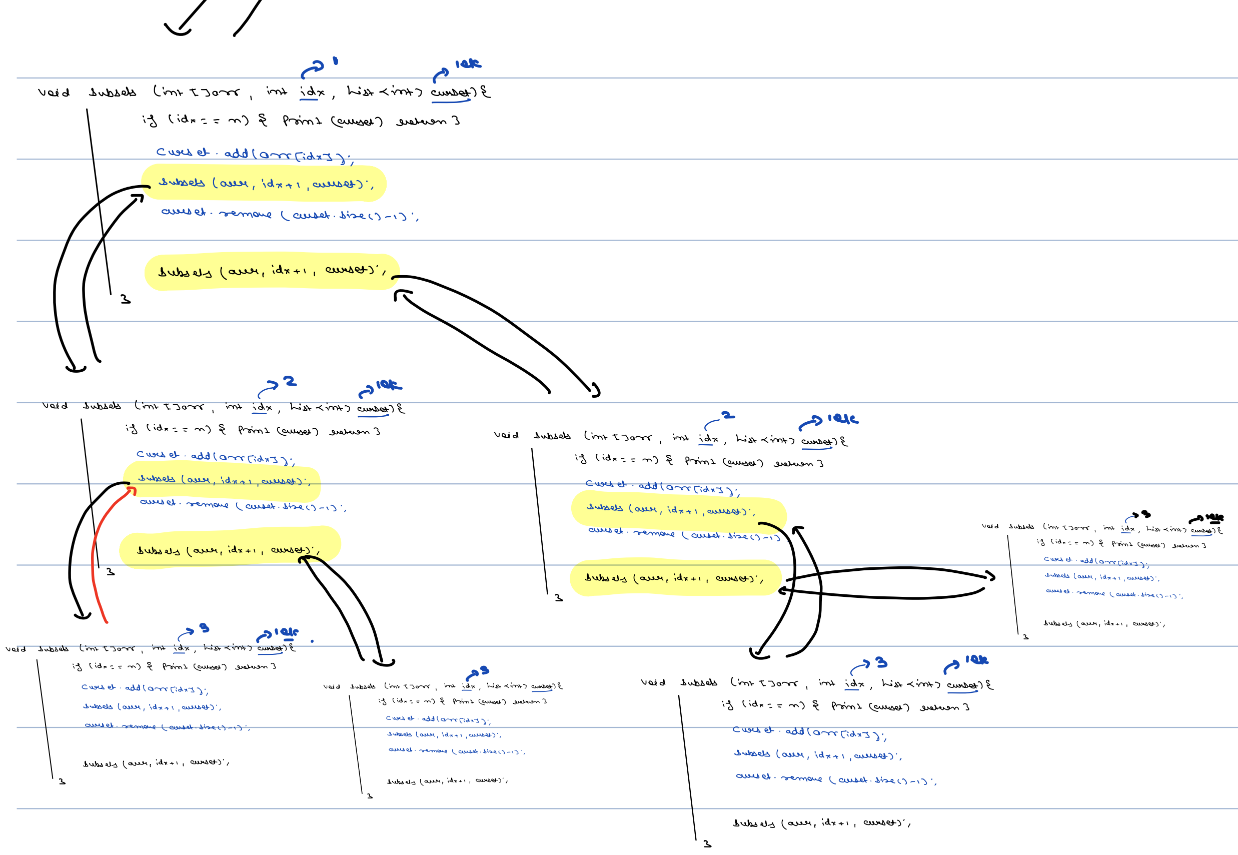
O/P

10,20,30
10,20
10,30
10

subsets([10,20,30],0,[]);

Currset → (   )

```
                                 →0             →10k
void  subsets (int []arr , int idx , List<int> currset){
        if (idx == n) { Print (currset)  return }
        currset . add(arr[idx]);
        subsets (arr, idx+1 ,currset);
        currset . remove (currset . size() -1);

        subsets (arr, idx+1, currset);
}
```



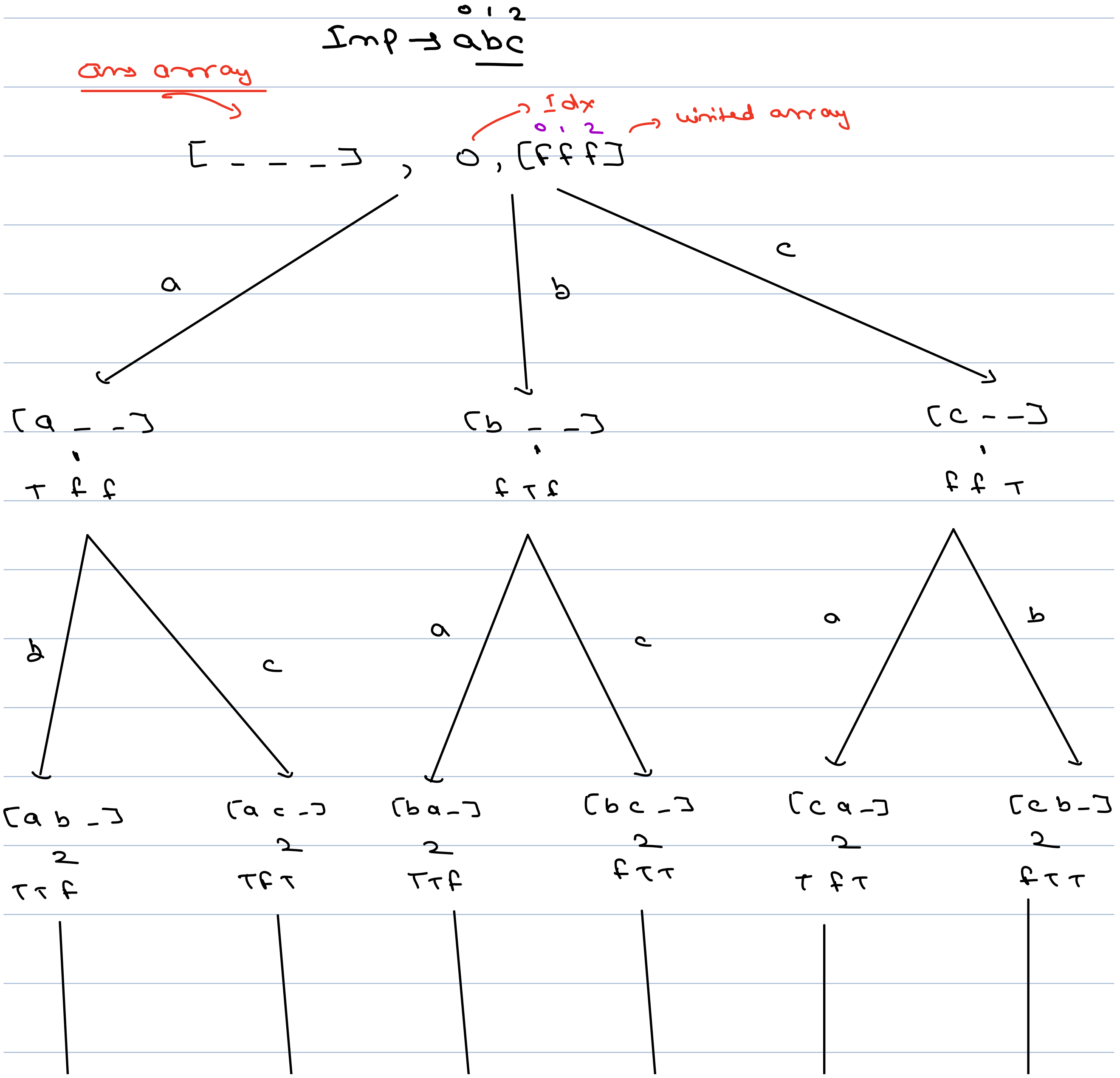
```
          →10k
list  →  (10,20,30,) 40

List<List<Int>  ans;      (10k)
        ans. add(list);
    list. add (40);
```

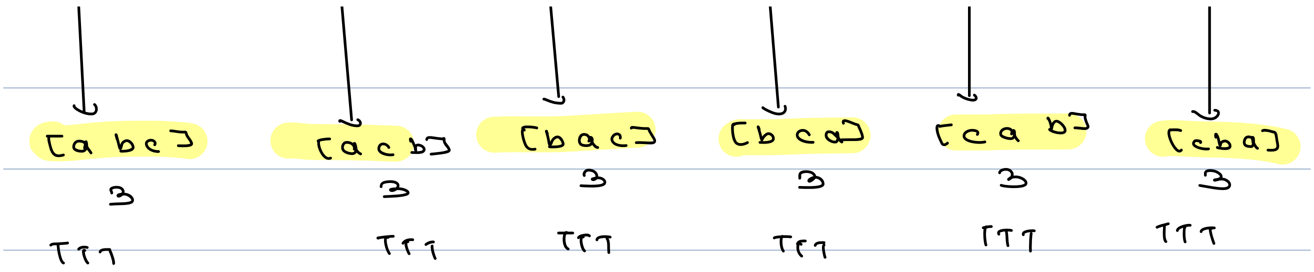
Ques Given a string with distinct characters, print all permutations.

(lowercase letters only)

abc → abc
acb
bca
bac
cab
cba

n → n! permutations





```
void  permutation (ans[], idx, vis[], str)
        if (idx == str.len) { print (ans); return }

            for (i -> 0 to n-1) {
                if (vis[i] == false) {
                    ans[idx] = str[i];
                    vis[i] = True;
                    permutation(ans, idx+1, vis, str);
                    vis[i] = false;
                }
            }
}
```

T.C → O(n * n!)

S.C → O(n).

kth Symbol hard .

0 → 01
1 → 10

n = 1   0

n = 2   0 1

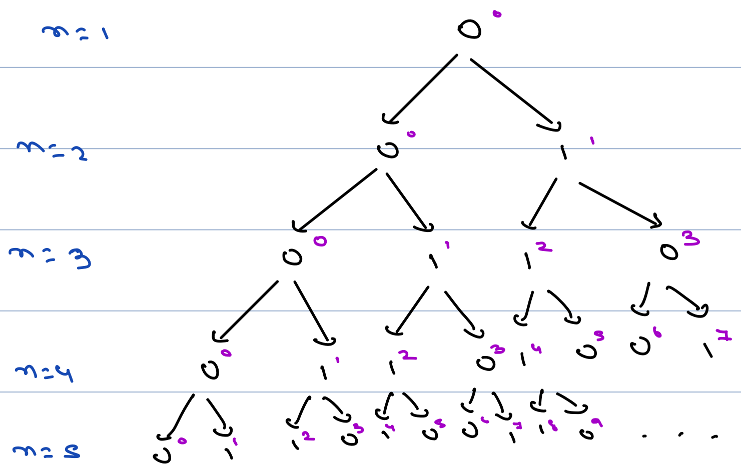
n = 3   0 1 1 0

n = 4   0 1 1 0 1 0 0 1

n = 5   0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0

(n = 5, k = 3) → 0

(n = 4, k = 6) → 0



obs 1) Every Even Idx element is same as its Parent .

obs 2) Every odd Idx element is opp. as its Parent .

obs 3) Parent Idx = $\frac{\text{MyIdx}}{2}$

obs 4) n = 1 , → 0 ,   k = 0 → 0

(n= 5, k= 6)

↓

(n= 4, k= 3)

↓

!(n= 3, k= 1)

↓

!(n=2, k=0)

(n= 5, k= 7)

↓

!(n= 4, k= 3)

↓

!(n= 3, k= 1)

↓

!(n= 2, k= 0)

```
int  kthSymbol ( n, k) {
      if(n==0||k==0) { return 0}
      if (k%2==0) {
              return  kthSymbol(n-1, k/2)
      }
      else {
          return  1-  kthSymbol(n-1, k/2)
      }
}
```