K^{th} element

Recursion 2 {HW}

N \ K	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
1	0																size = 1
2	0	1															size = 2
3	0	1	1	0													size = 4
4	0	1	1	0	1	0	0	1									size = 8
5	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0	size = 16
6																	

(N, k)

N , k

$N = 1 \quad k = 1 \longrightarrow 0$

$N = 3 \quad K = 2 \longrightarrow 1$

$N = 5 \quad k = 11 \longrightarrow 0$

$N = 10$

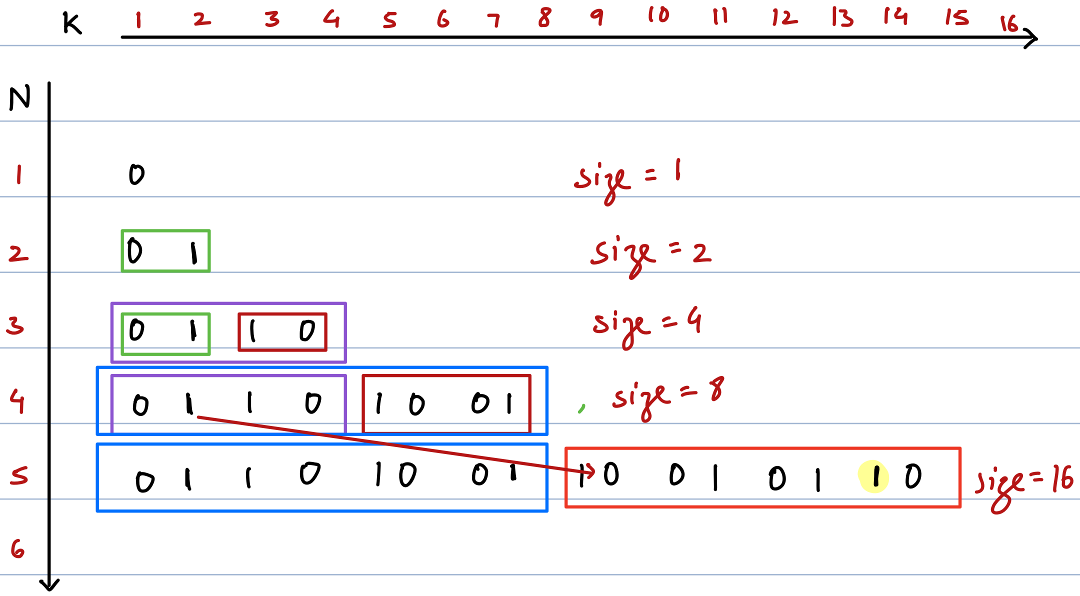
$Size = 2^{N-1}$

$\boxed{N = 100}$

$2^{99} \longrightarrow 10^{10}$

$1 \le N \le 10^{5}$

Optimised approach



N^{th} row → $(N-1)^{th}$ row

N = 5   k = 4

N^{th} row

previous row : (N-1)

size of prev row : 2^{N-2}

N = 5 → k ≤ 8   find ans from prev row

N = 5 → k > 8   find ans of k - 8 from previous row and compliment it

k = 10   complimenting   k = 2

k = 11   complimenting   k = 3

k = 15   "   " = 7

N^{th} row → $N-1^{th}$ row

Assumption: kth elemen (N, k) returns the (N, k) value from matrix

Main logic :

```
int kthelement (N,k){

    if (N==1) { return 0}

    prevrow size = pow(2, N-2)
    if (k <= prevrow size){
            # ans from prev row
            return kthelement (N-1, k)
    }
    else {     # k > prevrow size
            # negate n-1, k- prevrow size
            return 1- kthelement (N-1, k-prevrow size)
    }
}
```

Comparing 2 strings {lexiographically}

"a bc" < "d c"
↑ ↑

"abc" < "abd"
↑ ↑

"abcd" > "abc"
↑ ↑

s1 > s2 ⟶ 1

s1< s2 ⟶ -1

s1 == s2 ⟶ 0

"world"

"word"

for alien

```
int compare ( s1, s2) {
    int sz1 = s1.size()          3      cat
    int sz2 = s2.size()          3      elf
    for(i=0; i< min(sz1, sz2) ; i++){
        if ( hm[s1[i]] < hm[s2[i]] ){
            return -1
        }
        if (hm[s1[i]] > hm[s2[i]] ){
            return 1
        }
    }
    if (sz1 > sz2){
        return 1
    }
    else if (sz1 < sz2){
        return -1
    }
    else{
        retur 0
    }
}
```

l > d

index of s1[i] in order string
& index of s2[i] in order

Q2) Alien dictionary

```
          0 1 2 3 4 5 6 7
order = "w o r d a b c e f g h i j k l m n p q s t u v x y z"
                        r < e
                        ↓   ↓
                        2   7
```

```
l = [ "word" , "world", "row" , "elf" ]        Sorted
            <          <       <
```

unsorted

```
l = [ "word" , "world", "row" , "elf" , "erf" ]
                               <      >
```

A[i] < A[i-1]   for any i

not sorted in asc

```
for (i=1 ; i < N ; i++) {
    if (A[i] < A[i-1]) {
    |      return false
    }
}
return true
```

key : char [order[i]]

value : int [i]

```
hm = {}
for (i=0; i<26; i++) {
|     hm[order[i]] = i
}
```

```
for (i=1 ;  i<N; i++){
        s1 = l[i]
        s2 = l[i-1]
        int  c =  compare (s1, s2)
        if (c = = -1) {
                # s1 < s2
                return false
        }
}
return true
```

s2 s1

l = ["word" , "world", "row", "elf", "erf"]

↑

False

Add binary

10^5

A = "1010"

B = "11"

```
    1
1 0 1 0
    1 1
-------
1 1 0 1
```

```
if (A.size() < B.size()){
        s = "0" * (B.size - A.size)
        A = s + A
}
```

Ans : sum % 2

Cary : sum/2

```
          1   0
A = "1 0 1 1 0"
B = "0 0 0 1 1"
    -----------
            2 1
    -----------
Ans :       0 1
```

```
ans = ""
carry = 0
for (i= s1.size() -1 ; i>=0; i--){
    p1= int(s1[i])
    p2 = int(s2[i])
    sum = p1+p2 + carry
    carry = sum/2
    ans = ans + str(sum%2)
}
```

```
if (carry > 0){
    ans += str(carry)
}
```

```
rev(ans)
```

←

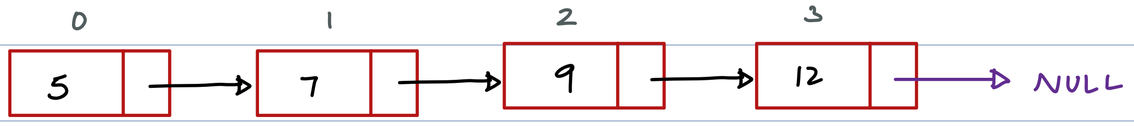
```
1           1
A:  1   1   0   1
B:  1   0   0   1
   --------------
    0   1   1   0
```

Break (10:40 - 10:50)

Q4) Insert at k^{th} position in linked list

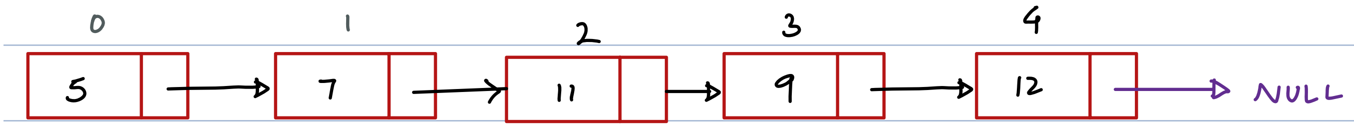
k will be valid value

$0 \le k \le N-1$

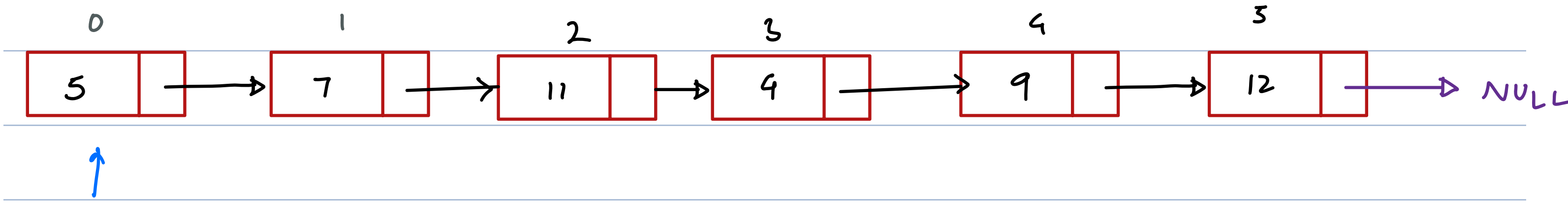


```
if (k==0){
    |          insertion at front
}
```

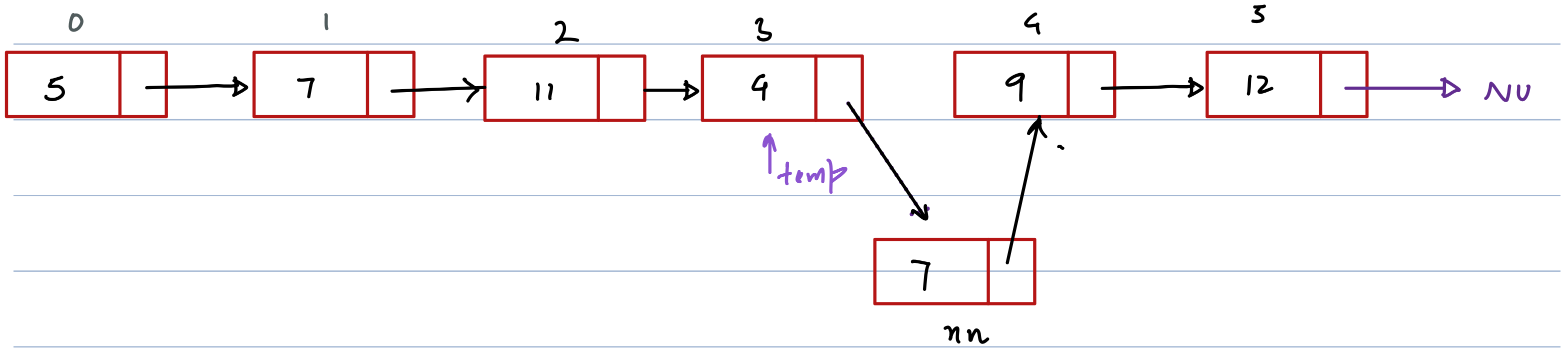
k=2 insert 11



k=3 insert 4



insert at k=3 you must be present at k=2



insert 7 at k=4

in order to insert at k^{th} index you must be present at $(k-1)^{th}$ index

$[0, k-1)$

```
for (i=0; i< k-1 ; i++){
        temp = temp.next
}

Node nn = new Node (7)
nn.next = temp.next
temp.next = nn
```

Q5)   max count = 2   ans = I   work

A = ["I" , "am" , "such" , "good" , "pgmr" , "pgmr" , "am" , "am" , "I"]

B = ["am" , "good"]   HS

1 word which occurs max no. of times in A but not present in B
if there are more than 1 words return lexicographically smalest

I	2
such	1
pgmr	2

Ans: I

Create HS for B

```
for (i=0; i<N; i++){
    if(B.contains(A[i])){
        continue
    }
    else{
        hm[A[i]]++
    }
}
```

max count = 0   ans = ""

```
for (i in hm) {
    key = i
    count = hm[i]
    if (count > max count) {
        ans = i
        maxcount = count
    }

    else if (count == max count) {
        if (i < ans) {
            ans = i
        }
    }
}

return ans
```