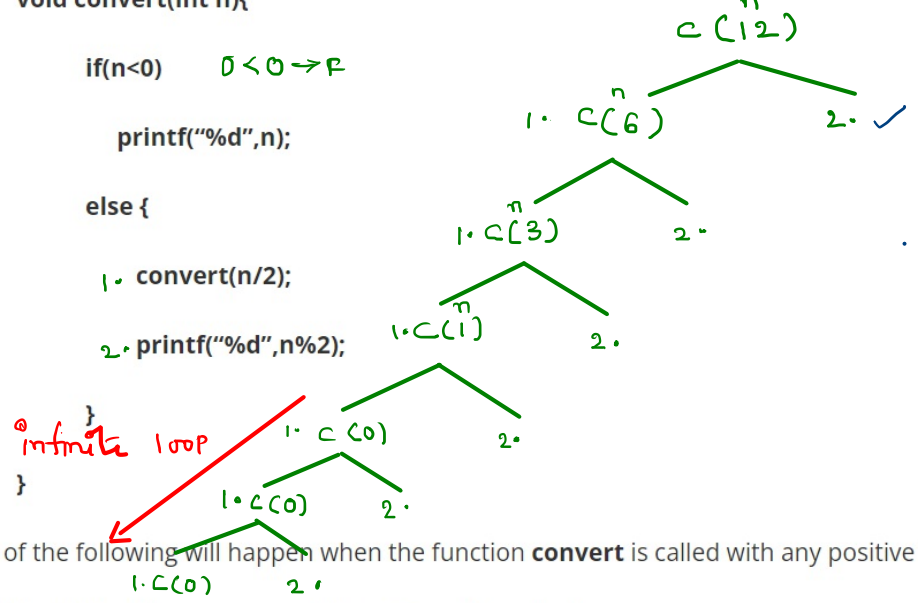
Day-10

$n = 12 \rightarrow (1100)_2$

$n = 13 \rightarrow (1101)_2$

Consider the following C function.

```
void convert(int n){
    if(n<0)
        printf("%d",n);
    else {
        convert(n/2);
        printf("%d",n%2);
    }
}
```



⇒ Base case is not correct

∴ Recursion → B.C
↳ mandatory

Which one of the following will happen when the function **convert** is called with any positive integer **n** as argument?

- (A) It will print the binary representation of **n** and terminate
- (B) It will print the binary representation of **n** in the reverse order and terminate
- (C) It will print the binary representation of **n** but will not terminate
- (D) It will not print anything and will not terminate

1)Find the first Occurance of target in the given array

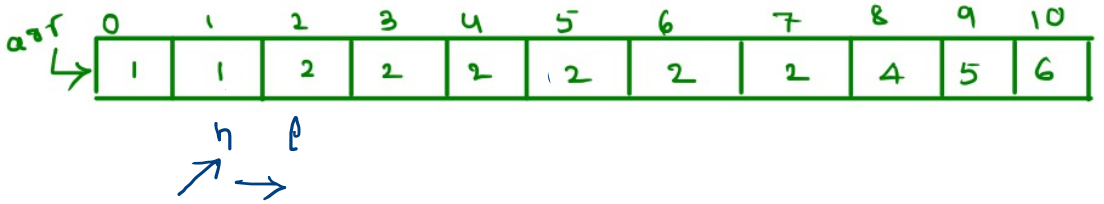
arr [] = { 1, 1, 2, 2, 2, 2, 2, 2, 4, 5, 6 }, target=2 → o/p (2)

↳ index

↳ Linear search → O(n)

first B.S

key = 2



res = ~~-1~~ 2

key Vs arr[mid]

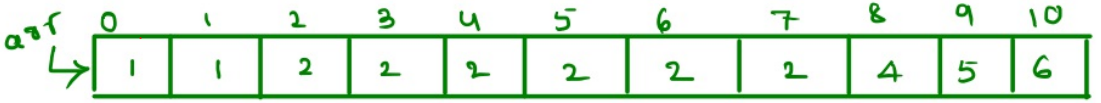
==

>

<

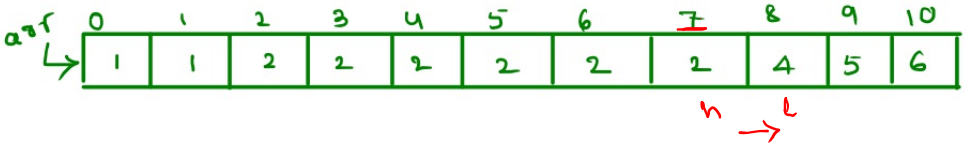
```
function first(arr[],n,key)
{
        low=0,high=n-1,res=-1
        while(low<=high)
       {
            mid=low+(high-low)/2
            if(arr[mid]==key)
            {
                 res=mid
                 high=mid-1
            }
            ese if(key>arr[mid])//R.H.S
                low=mid+1
           else //L.H.S
              high=mid-1
       }
       return res

}
```


arr
0 1 2 3 4 5 6 7 8 9 10
1 1 2 2 2 2 2 2 4 5 6

2)Find the last Occurance of target in the given array

arr [] = { 1, 1, 2, 2, 2, 2, 2, 2, 4, 5, 6 }, target = 2




```
function first(arr[],n,target)
{

}
```

Key is present

res = ~~-1~~

8

6 7

```
function last(arr[],n,key)
{
        low=0,high=n-1,res=-1
        while(low<=high)
       {
             mid=low+(high-low)/2
             if(arr[mid]==key)
             {
                   res=mid
                   low=mid+1
             }
            else if(key>arr[mid])//R.H.S
                low=mid+1
            else //L.H.S
              high=mid-1
       }
       return res
}
```

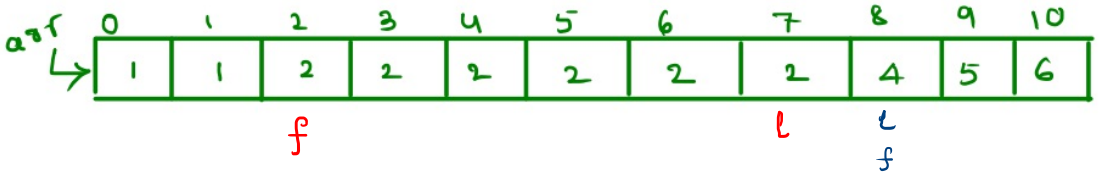
3)Find the frequency of target in the given array

① x = first(arr, 0, n-1, target) → O(logn)

② y = last(arr, 0, n-1, target) → O(logn)

→ arr [] = { 1, 1, 2, 2, 2, 2, 2, 2, 4, 5, 6 }, target=2 ⇒ freq(2) = 6

⇒ O(log n)



⇒ element is not present

l = -1

f = -1

l - f + 1 = -1 + 1 + 1

= (1)

* l - f + 1 = 7 - 2 + 1

= (6)

⇒

```
if( x == -1 and y == -1)
{
    return -1;
}
else
{
    return y-x+1;
}
```

Find the floor and ceil of a element in a given array

arr → 0 1 2 3 4 5 6
1 2 8 10 10 12 19

Given a sorted array and a value x, the ceiling of x is the smallest element in array greater than or equal to x, and the floor is the greatest element smaller than or equal to x. Assume than the array is sorted in non-decreasing order. Write efficient functions to find floor and ceiling of x.

Examples :

```
For example, let the input array be {1, 2, 8, 10, 10, 12, 19}
For x = 0:    floor doesn't exist in array,  ceil  = 1
For x = 1:    floor  = 1,  ceil  = 1
For x = 5:    floor  = 2,  ceil  = 8
For x = 20:   floor  = 19,  ceil doesn't exist in array
```

floor ← x → ceil
LHS RHS

ceil(7) = ?
→ 8, 10, 12, 19
→ 8

ceil(2) = 2

ceil(21) = -1

floor(13) = ?
→ 12, 10, 8, 2, 1
→ 12

key = 13 < f = 5 (12)
c = 6 (19)

arr

0	1	2	3	4	5	6	7
1	2	8	10	10	12	19	31

h l

X

- f = 5 ✓
- c = 6 ✓

Initial set-up

f = -1

c = -1

l = 0

h = 7

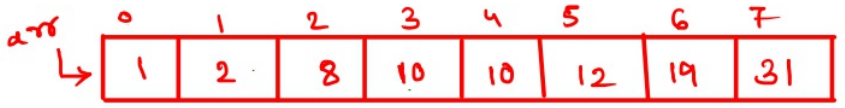
13

key V/s arr[mid]

== → f = mid, c = mid
break

> → low = mid + 1 ✓
f = mid

< → high = mid - 1 ✓
c = mid




```
function fun(arr[],n,key)
{
   low=0, high=n-1, f=-1, c=-1
   while(low<=high)
  {
       mid=low+(high-low)/2
       if(key==arr[mid])
       {
           f=mid
           c=mid
           break
       }
       if(key>arr[mid])//R.H.S
       {
          f=mid
          low=mid+1
       }
       else //L.H.S
       {
            c=mid
             high=mid-1
       }
   }
   print(f)
   print(c)

}
```

Search for an element in an infintely sorted array

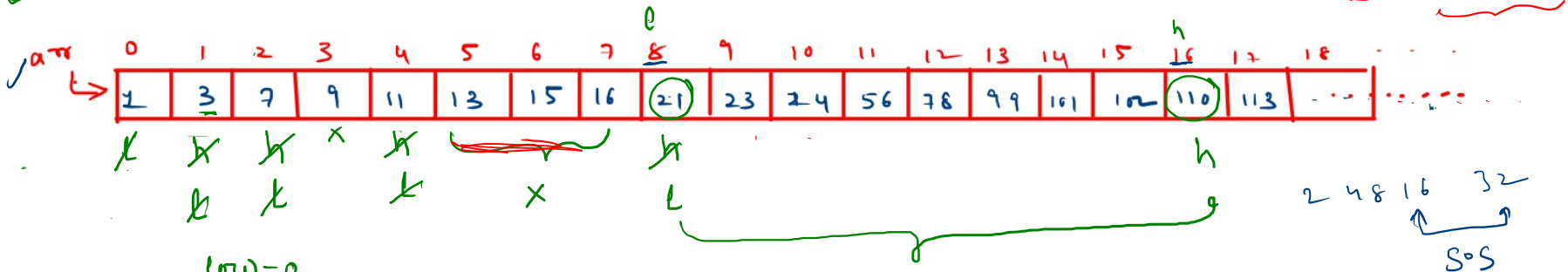
$O(n) \rightarrow O(\log_2 n)$

Key = 78 ⇒ B.S(?)

→ Sorted

Key = 1000

500 1500



2 4 8 16 32

S.S

→ BS(arr, 8, 16, 78)

Step 1

```
low = 0
high = 1
while ( key > arr[high] )
{
    low = high
    high = 2 * high
}
```

Step 2 ⇒