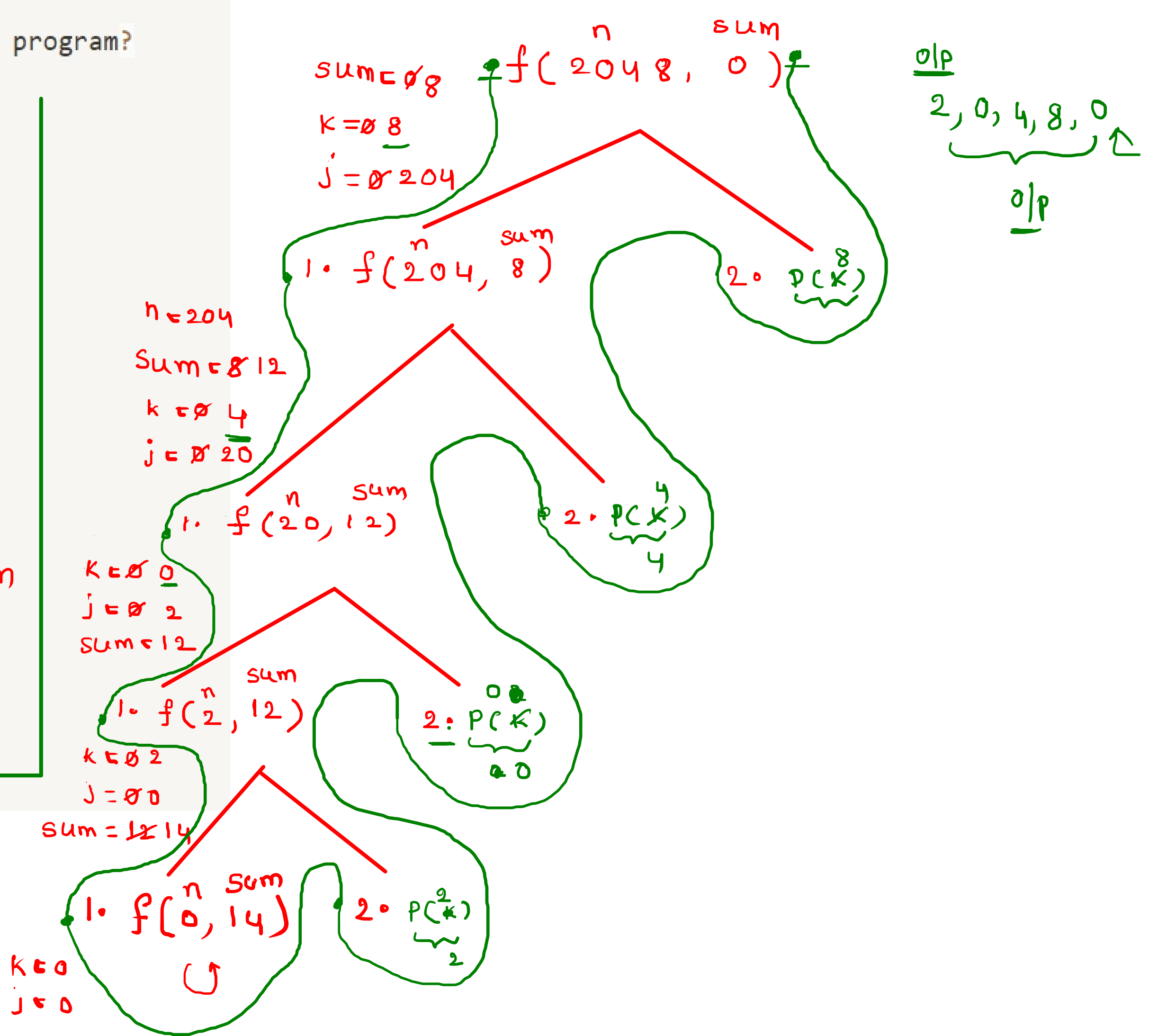
Day-9

what is the output for the following program?

```
function fun(n,sum)
{
    k=0,j=0
    if(n==0)
       return;
    k=n%10,j=n/10
    sum=sum+k
 1. fun(j,sum)
 2. print(k)
}
main()
{
    a=2048,sum=0
    fun(a,sum)
    print(sum)
}
```

a: 2048   sum: 0



→ file[txt, arr, ...], Key

** Binary Search

0 False

1 True

→ pre-condition :- Array must be sorted.

1. Linear search → loop → O(n)

↳ arr[ ]

↳ elements present in arr are random order

B.S ✓

Key = 27, $n = 9$ ✓

$n \leftarrow$ [search-space]

n

$\downarrow$

$n/2$

$\downarrow$

$n/4$

⋮

1

arr

0	1	2	3	4	5	6	7	8
2	4	8	12	16	19	21	27	32

✓ sorted.

l m h

?

Key v/s arr[m]

o/p

C_1:- Key == arr[m] $\rightarrow$ m

C_2:- Key > arr[m] $\rightarrow$ traverse on RHS
l = mid+1

C_3:- Key < arr[m] $\rightarrow$ traverse on LHS
h = mid-1

① $m = \dfrac{l+h}{2}$

Implementation

↳ ② $m = l + \dfrac{(h-l)}{2}$ * $\Rightarrow \dfrac{2l+h-l}{2} = \dfrac{l+h}{2}$

BS
DSA

$m = l + \frac{(h-l)}{2}$ $\Rightarrow$ array $\rightarrow$ type integer

$\downarrow$

Let 4 Bytes

1 Byte = 8 bits

$\downarrow$

32-bit

$\downarrow$

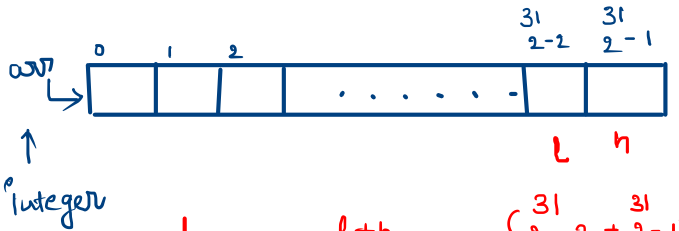
$(-2^{31}$ to $2^{31}-1)$

n
4-bit
$\downarrow$
-2^{3} to $2^{3}-1$

-8 to 7

2's comp

n-bit
$\downarrow$
-2^{n-1} to $2^{n-1}-1$



$^*m = l + \frac{(h-l)}{2}$

$= 2^{31}-2 + \frac{(2^{31}-1-2^{31}+2)}{2}$

$m = 2^{31}-2$ $\quad = 2^{31}-2 \rightarrow \frac{1}{2} \rightarrow 0$

$m = \frac{l+h}{2}$ $\Rightarrow$ $(2^{31}-2+2^{31}-1)$

$\downarrow$

$2 \cdot 2^{31} - 3$

$= [2^{32} - 3]/2$

over flow

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

}

main()
{
     arr[] = { 1, 4, ,7 ,8, 23,36 }
     index=binarySearch(arr,0,arr.length-1,8)
     if(index==-1)
           print(element not found)
     else
          print(element found)
}
```

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

}
```

**

$$T(n) = \begin{cases} 1 & ; n=1 \\ 1 + T(n/2) & ; n \geq 2 \end{cases}$$

Time and Space Complexity Analysis of Binary Search

$n \downarrow n/2 \downarrow n/4 \downarrow n/8 \ldots 1$

Let $T(n)$: be the T.C of Binary Search with n elements

\# of elements

$$T(n) = \begin{cases} 1 & ; n=1 \text{ (Base condition)} \\ 1 + T(n/2) & ; n \geq 2 \end{cases}$$

→ Back-substitution

$T(n) = 1 + T(n/2) \rightarrow (1)$

$T(n/2) = 1 + T(n/4) \rightarrow (2)$

Substitute (2) in (1)

$= 1 + 1 + T(n/4) = 2 + T(n/2^2) \rightarrow (3)$

$T(n/4) = 1 + T(n/8) \rightarrow (4)$

substitute (4) in (3)

$= 2 + 1 + T(8) = 3 + T(n/8)$

$\vdots$ } after k steps

$= k + T(n/2^k)$

$= k + T(n/2^k)$

$* \quad \frac{n}{2^k} = 1 \Rightarrow n = 2^k$

Apply $\log_2$ on Both sides

$\log_2 n = \log_2 2^k \Rightarrow \log_2 n = k \cdot \underbrace{\log_2 2}_{1}$

$$\boxed{k = \log_2 n}$$

$= \log_2 n + T(1)$

$= (\log_2 n + 1) \Rightarrow O(\log_2 n)$

Assume n is a perfect square $\sqrt{n} \Rightarrow ?$ $n = 625 \longrightarrow$ O/P 25 . BF

$n = 144 \longrightarrow 12$

find square root of number :-

n^2

1

l

m

625

h

$i = \cancel{1} \Rightarrow i*i$

O(n)

l

l = 1	1	1	1	1	20	20	25	25
h = 625	312	155	77	38	38 h	28	28	25
m = 313	156	78	39	19	29	24	26	Stop

(mid * mid)   19*19   25*25

log(log n)

$\cancel{1}$ $\cancel{2}$ $\cancel{3}$ $\cancel{4}$ $\cancel{5}$ $\cancel{6}$ $\cancel{7}$ $\cancel{8}$ $\cancel{9}$ $\cancel{10}$ $\cancel{11}$

n = 144 $\sqrt{n}$ ✓

$\downarrow, \sqrt{n} \Rightarrow O(\sqrt{n})$

12 times

T.C ✓

S.C ✓

Basic stuff

Binary Search $\Rightarrow O(\log_2 n)$ → large values

Assymptotically

↳ Consider for the larger values of (n)

$12 \Rightarrow 12*12 = 144$

$n = 400$

h

ℓ

1