Q1. Given a no. write recursive program to calculate sum of digits of that no.

eg ⇒ N = 42689 , sum(N) = 4+2+6+8+9
= 29 → o/p.

// assumption

sumDigit(N) → returns sum of all digits of no. N.

// main logic

$\text{sumDigit}(42689) = \underbrace{9}_{\text{last digit}} + \text{sumDigit}(\overbrace{4268}^{\text{except the last digit}})$

N % 10 ⇒ last digit of N

N / 10 ⇒ remove the last digit of N

100 % 10 = 0

55 % 10 = 5

62 % 10 = 2

1234 / 10 ⇒ 123

5670 / 10 ⇒ 567

⇒ return N%10 + sumDigit(N/10)

// base condn :

```
if( N <= 9)
    return N
```

pseudo [TODO]

Q2. Implement power function. given a, N return a^N.

eg ⇒ $a = 3$, $N = 3$, $3^3 \Rightarrow 27$.

$$\underbrace{a^N = a \times a^{N-1}}_{\text{recursive}} \qquad a^0 = 1.$$

pseudo

```
int pow(a, n) {
        if(n == 0)
            return 1
    return a * pow(a, n-1)
}
```

a^5
↓
$a \times a^4$ ✓
↓
$a \times a^3$ ✓
↓
$a \times a^2$ ✓
↓
$a \times a^1$ ✓
↓
$a \times a^0$ ✓

$\underbrace{\text{5 multiplications}}$

$a^N \rightarrow N$ multiplications

iterative

```
result = 1
for( i=1; i<=N; i++) {
        result = result * a
}
```

⇒ N multiplications

$a^{10} \Rightarrow a^9 \times a$

$a^{10} \Rightarrow a^5 \times a^5$

$a^{14} \Rightarrow a^7 \times a^7$

$a^{15} \Rightarrow a^7 \times a^7 \times a$

$a^{20} \Rightarrow a^{10} \times a^{10}$

$a^{21} \Rightarrow a^{10} \times a^{10} \times a.$

✓ $a^N = a^{N/2} \times a^{N/2}$, if $N \% 2 == 0$

✓ $a^N = a^{N/2} \times a^{N/2} \times a$, if $N \% 2 \; != 0.$

$\Rightarrow a^{64} \rightarrow 64$ multiplications

$\Rightarrow a^{64} \longrightarrow a^{32} \times a^{32}$

$\downarrow$

$a^{16} \times a^{16}$

$\downarrow$

$a^8 \times a^8$

$\downarrow$

$a^4 \times a^4$

$\downarrow$

$a^2 \times a^2$

$\downarrow$

$a^1 \times a^1$

6 multiplications

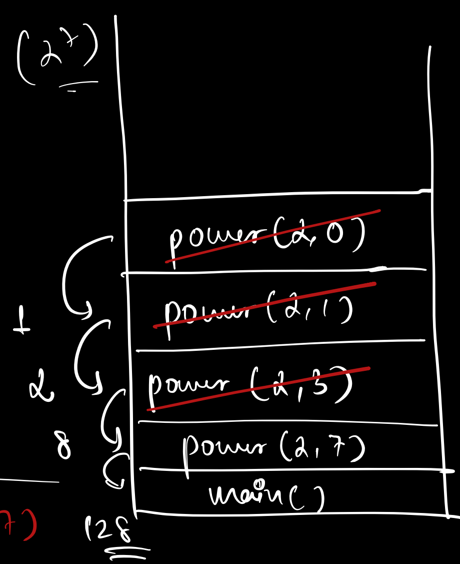
N is divided by 2 till reached 1 ⇒ log N

pseudo

```
int power(a, n){
        if(n==0)
            return 1;
    int halfpower = power(a, n/2);
    if(n%2==0){
        return halfpower x halfpower
    } else {
        return halfpower x halfpower x a
    }
```


```
int power(a, n){
        if(n==0)
            return 1;
    int halfpower = power(a, n/2);
    if(n%2==0){
        return halfpower x halfpower
    } else {
        return halfpower x halfpower x a
    }
```


O/p = 128 (2^7)



pow(2,1)	pow(2,3)	pow(2,7)
halfpower = 1	hp = 2	hp = 8
return ⇒ 1*1*2	2*2*2	8*8*2 = 128

pow(2,8)

pow(2,8)

hp = 16

N = 8

hp * hp

16 * 16 = 256

pow(2,0)

1 pow(2,1)

2 pow(2,2)

4 pow(2,4)

16 pow(2,8)

256

$a^{10} \Rightarrow a^{10/3} \times a^{10/3} \times a^{10/3} \times a$

$a^{10} \Rightarrow a^{10/4} \times a^{10/4} \times a^{10/4} \times a^{10/4} \times a^{2}$

/2

100 → 50 → 25 → 12 → 6 → 3 → 1

$\log_2 N$

/3

100 → 33 → 11 → 3 → 1

$\log_3 N$

if(2)

~~else~~ else if (1)

else

$a^N \to a^{N/2} \times a^{N/2} \to \log_2 N$

$a^N \to a^{N/3} \times a^{N/3} \times a^{N/3} \to \log_2 N$

$a^N \to a^{N/4} \times a^{N/4} \times a^{N/4} \times a^{N/4} \to \log_4 N$

$a^N \to \left(a^{N/N} \times a^{N/N} \times a^{N/N} \times a^{N/N} \cdots \times a^{N/N}\right)$ N times

N times $(\log_N N) \to O(N)$

$a^{N/N}$

$= a^1 \left[a \times a \times a \times a \times a \cdots\right]$ N times

iterative

$\Rightarrow$ return $a^N \% d$

ex $\Rightarrow$ $a = 2$ $\quad d = 10^9 + 7$

$N = 10$

$(2^{10})\ \%\ d \Rightarrow (2^{10})\ \%\ 10^9+7$

```
int power(a, n){
    if(n==0)
        return 1;
  int halfpower = power(a, n/2);
  if( n%2==0){
      return (halfpower%d x halfpower%d )%d
  }
  else {
      return (halfpower%d x halfpower%d x a%d)%d
  }
```

modulo multiplicative rule

$(a \times b)\%d$

$= (a\%d \times b\%d)\%d$

Time complexity:

$$sum(N) \rightarrow N + sum(N-1) \rightarrow (N-1) + sum(N-2) \rightarrow (N-2) + sum(N-3) \rightarrow \cdots sum(N)$$

N times add^n

$TC \Rightarrow O(N)$

$$SUM(N) \longrightarrow T(N)$$

$$SUM(N-1) \rightarrow T(N-1)$$

$$\begin{array}{ccc} SUM(N) = & \underbrace{N +} & SUM(N-1) \\ \downarrow & & \downarrow \\ T(N) = & 1 + & T(N-1) \end{array}$$

$$T(N) = 1 + \underbrace{T(N-1)}$$

$$= 1 + (1 + T(N-2))$$

$$= 2 + \underbrace{T(N-2)}$$

$$= 2 + (1 + T(N-3))$$

$$= 3 + T(N-3)$$

$$\rightarrow T(N) = 1 + T(N-1)$$

$$T(N-1) = 1 + T(N-1-1)$$

$$= \underline{\underline{1 + T(N-2)}}$$

$$T(N-2) = 1 + T(N-2-1)$$

$$= \underbrace{1 + T(N-3)}$$

k th step

$$T(N) = k + T(N-k)$$

$$\downarrow$$

$$T(N-k) = T(0)$$

$$\Rightarrow N-k = 0$$

$$\Rightarrow k = N$$

after N steps

$$T(N) = k + T(N-k)$$

$$T(N) = N + T(N-N)$$

$$= N + \text{const}$$

$$\boxed{T(N) \Rightarrow O(N)}$$

$T(N) = 2T(N-1) + 1$

$$\Rightarrow 2^1 T(N-1) + 2^1 - 1$$

$$= 2\left[2T(N-2) + 1\right] + 1$$

$$= 4T(N-2) + 3$$

$$\Rightarrow 2^2 T(N-2) + 2^2 - 1$$

$$= 4\left[2T(N-3) + 1\right] + 3$$

$$= 8T(N-3) + 7.$$

$$= 2^3 T(N-3) + 2^3 - 1$$

$$T(N) = 2T(N-1) + 1$$

$T(N-1)$

$$= 2T(N-1-1) + 1$$

$$= 2T(N-2) + 1$$

$T(N-2)$

$$= 2T(N-2-1) + 1$$

$$= 2T(N-3) + 1$$

kth step $\quad T(N) = 2^k T(N-k) + 2^k - 1$

at end

$$T(N-k) = T(0)$$

$$\Rightarrow \underbrace{N = k}$$

$$T(N) = 2^k \, T(N-k) + 2^k - 1$$

$$T(N) = 2^N \, T(N-N) + 2^N - 1$$

$$= 2^N \, \underbrace{T(0)} + 2^N - 1$$

$$= 2^N + 2^N - 1$$

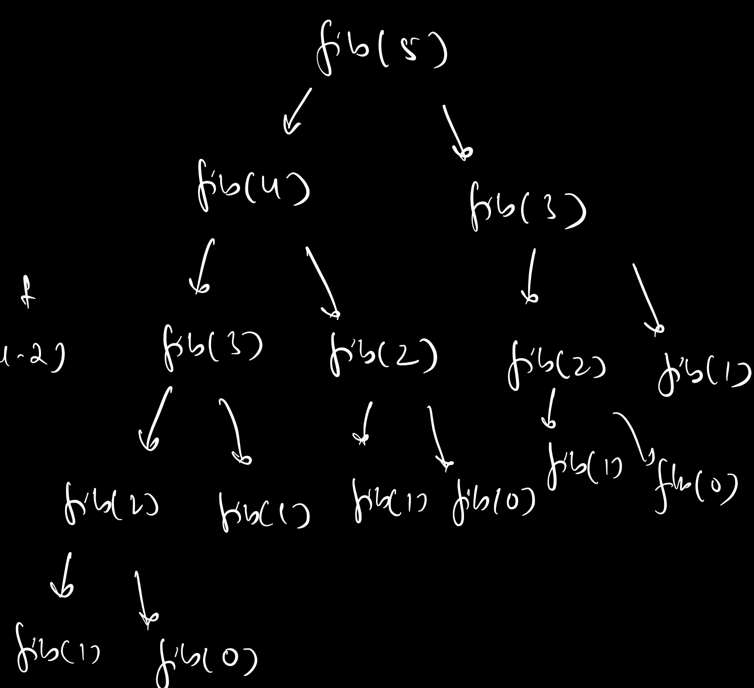
$$= 2 * 2^N - 1$$

$$\boxed{T(N) \Rightarrow O(2^N)}$$

```
int fib(N){

    if( N==0)
          return 0
    else if( N==1)
          return 1
    return fib(N-1) + fib(N-2)
}
```

$$\boxed{TC \Rightarrow O(2^N)}$$



$fib(N) = fib(N-1) + fib(N-2)$

$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$

$T(N) \qquad T(N-1) \qquad T(N-2)$

approx

$2T(N-1)+1$

$$\boxed{T(N) = 2T(N-1)+1}$$

$\Rightarrow$

pow(a, n)

$\rightarrow$ 2 pow(a, n/2)

$TC = O(N)$

```
int power(a, n){
    if(n==0)
        return 1;
    if(n%2==0){
        return power(a,n/2) x power(a,n/2)
    }
    else {
        return power(a,n/2) x power(a,n/2) x a
    }
}
```

$pow(a,n) = pow(a,n/2) \times pow(a,n/2)$ — —

$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$

$T(N) \qquad T(N/2) + T(N/2) + 1$

$$T(N) = 2\underbrace{T(N/2)} + 1$$

$$\Rightarrow 2^1 T(N/2) + 2^1 - 1$$

$$= 2\left[2T(N/4) + 1\right] + 1$$

$$= 4\underbrace{T(N/4)} + 3.$$

$$\Rightarrow 2^2 T(N/2^2) + 2^2 - 1$$

$$= 4\left[2T(N/8) + 1\right] + 3$$

$$= 8T(N/8) + 7$$

$$\Rightarrow 2^3 T(N/2^3) + 2^3 - 1$$

$$T(N) = 2T(N/2) + 1$$

$$T(N/2) = 2T\left(\frac{N/2}{2}\right) + 1$$

$$= \underbrace{2T(N/4) + 1}$$

$$T(N/4) = 2T\left(\frac{N/4}{2}\right) + 1$$

$$= 2T(N/8) + 1$$

kth

$$T(N) = 2^k T(N/2^k) + 2^k - 1$$

at end

$$T(N) = 2^{\log_2 N} T\left(\frac{N}{2^{\log_2 N}}\right) + 2^{\log_2 N} - 1$$

$$T\left(\frac{N}{2^k}\right) = T(1)$$

$$\Rightarrow \frac{N}{2^k} = 1$$

$$\Rightarrow 2^k = N$$

$$\Rightarrow k = \underline{\log_2 N}$$

$$T(N) = N + T\left(\frac{N}{N}\right) + N - 1$$

$$= 2N - 1$$

$$\boxed{O(N)}$$

$$a^{\log_a N} = N$$