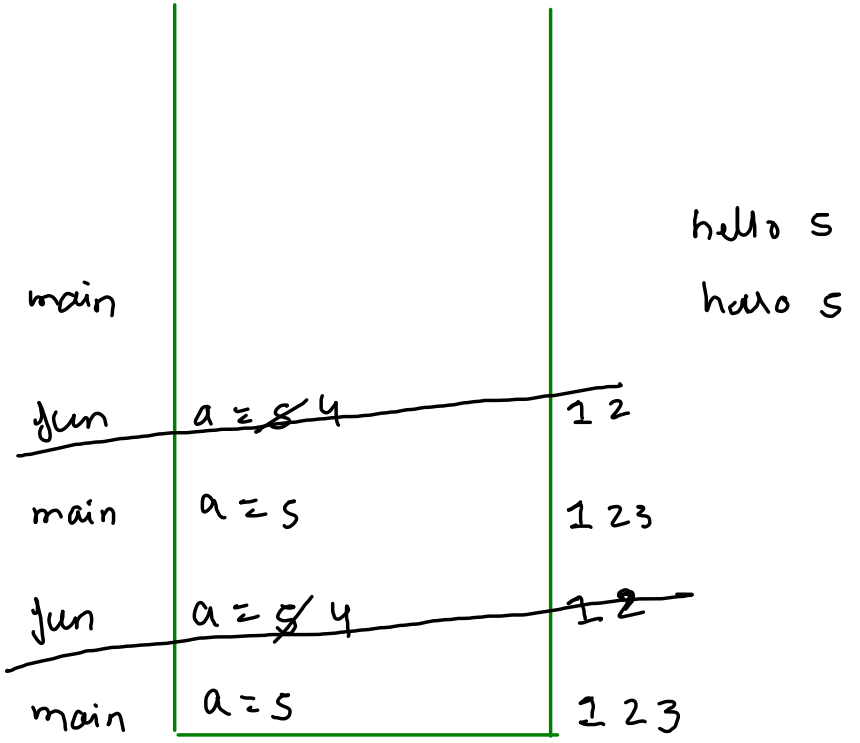
Recursion

```
void main() {
1. int a = 5;
2. fun(a);
3. main();
}

void fun(int a) {
1. syso("hello" + a);
2. a--;
}
```



PMI

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \quad -(1) \qquad i \in N$$

$$\sum_{i=1}^{1} = \frac{1(1+1)}{2}$$

$$1 = 1$$

(i) assume that eq. 1 is true for $n = k$

(ii) prove that formula is also for $n = k+1$

(iii) prove that formula is true for the smallest problem

1 2 2 3 4

$$\sum_{i=1}^{n=k} i = \frac{k(k+1)}{2} \quad -(2)$$

$$\frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n=k+1} i = \frac{(k+1)(k+1+1)}{2} = \frac{(k+1)(k+2)}{2}$$

$$1 + 2 + 3 + \cdots + k + (k+1) =$$

$$\frac{k(k+1)}{2} + (k+1) = \frac{k(k+1) + 2(k+1)}{2}$$

$$= \frac{(k+1)(k+2)}{2}$$

Recursion

Factorial

expectation $fact(n) = n * (n-1) * (n-2) * \ldots\ldots 1$

faith $fact(n-1) = (n-1) * (n-2) * \ldots\ldots 1$

meeting expectation using faith

$$fact(n) = n * fact(n-1);$$

1. decide your expectation.
2. freeze your faith.
3. meet expectation using faith.

Print decreasing

expectation pd(n) → n n-1 n-2 n-3 1

faith pd(n-1) → n-1 n-2 n-3 1

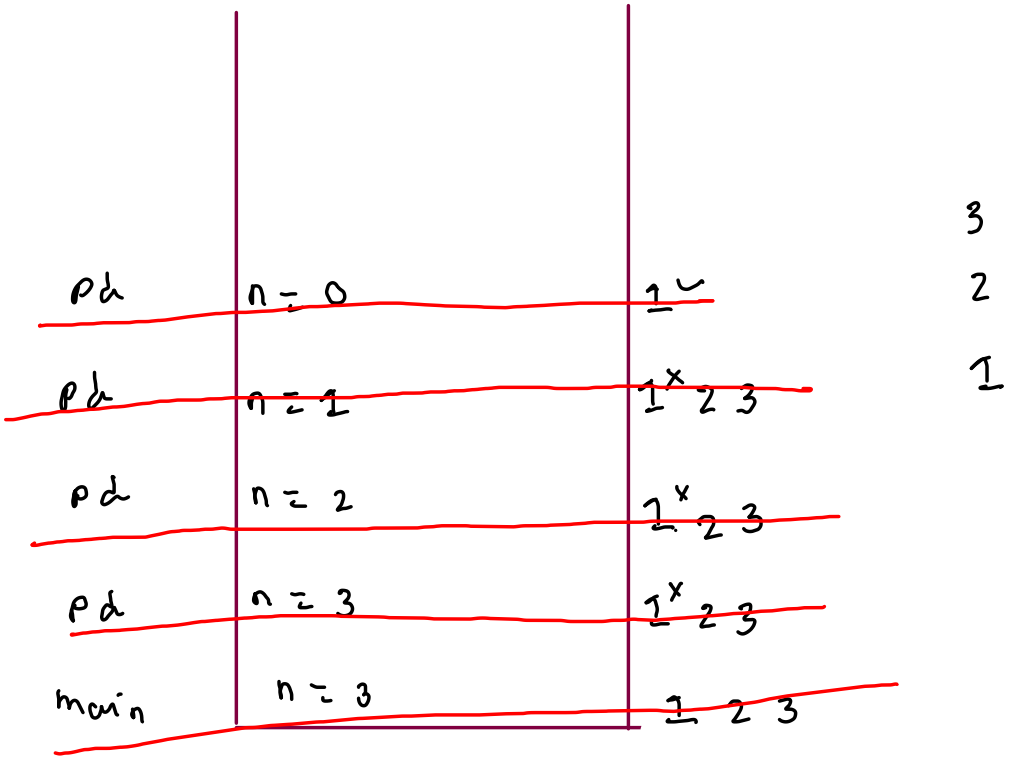
pd(n) → syso(n) + pd(n-1)

n n-1 n-2 n-3 ... 1 n n-1 n-2 n-3 ... 1

```java
public static void main(String[] args) throws Exception {
    // write your code here
1   Scanner scn = new Scanner(System.in);
2   int n = scn.nextInt();

3   printDecreasing(n); //expectation -> n n-1 n-2 n-3...1
}

public static void printDecreasing(int n){
1   if(n == 0) {
        return;
    }

2   System.out.println(n);

    //faith -> n-1 n-2 n-3...1
3   printDecreasing(n-1);
}
```

5 4 3 2 1

```java
public static void main(String[] args) throws Exception {
    // write your code here
1   Scanner scn = new Scanner(System.in);
2   int n = scn.nextInt();

3   printDecreasing(n); //expectation -> n n-1 n-2 n-3...1
}

public static void printDecreasing(int n){
1   if(n == 0) {
        return;
    }

2   System.out.println(n);

    //faith -> n-1 n-2 n-3...1
3   printDecreasing(n-1);
}
```



print 9nc

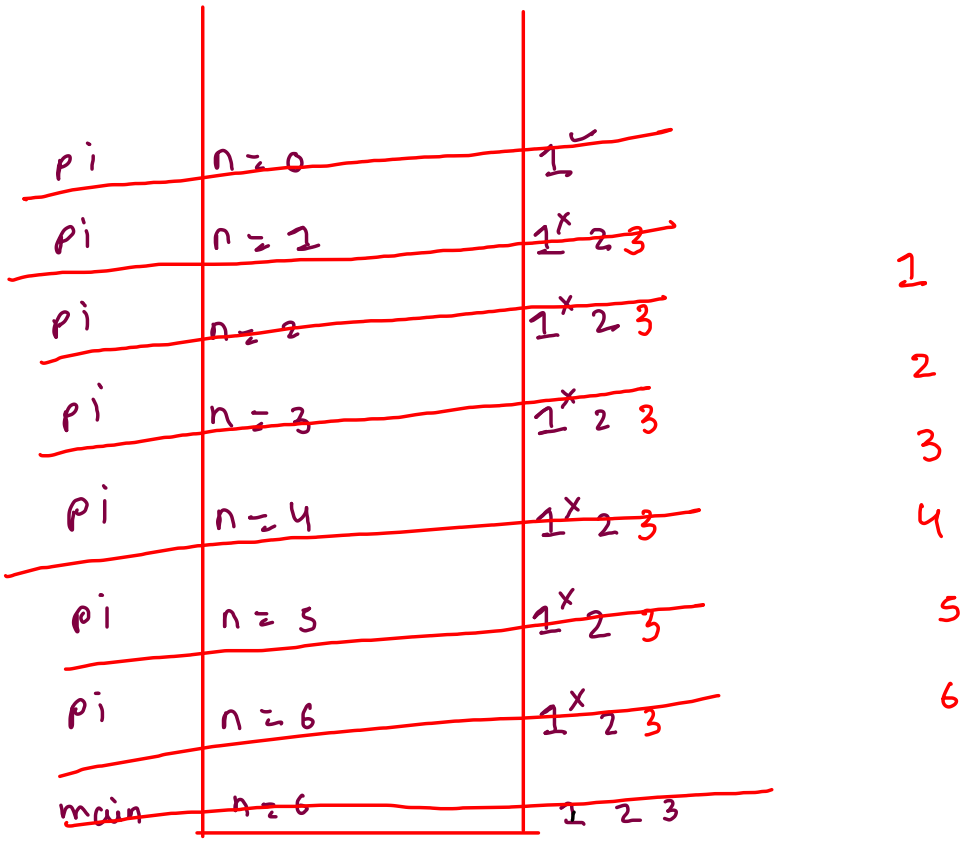
pi(n) → 1 2 3 4 5 ..... n-2 n-1 n

pi(n-1) → 1 2 3 4 5 ..... n-2 n-1

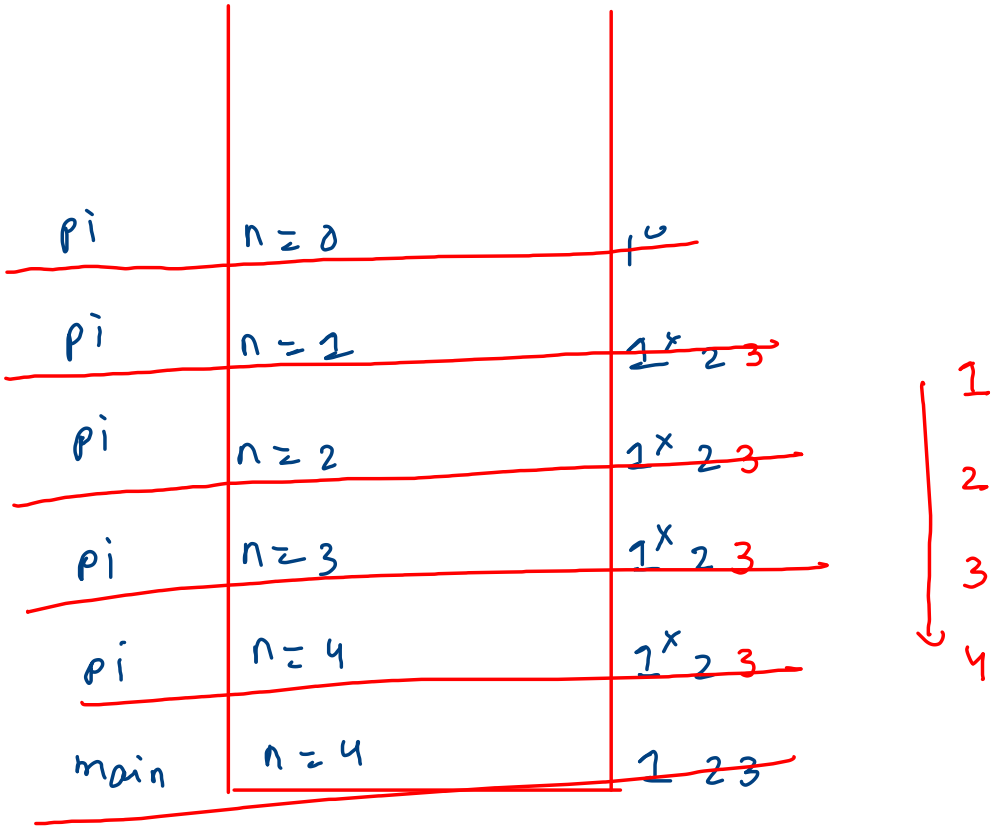
pi(n) = pi(n-1) + syso(n)

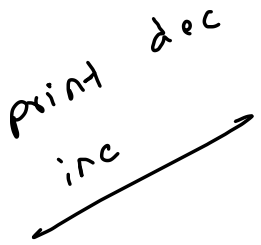
1 2 3 ..... n-2 n-1 n

```java
public static void main(String[] args) throws Exception {
    // write your code here
1   Scanner scn = new Scanner(System.in);
2   int n = scn.nextInt();

3   printIncreasing(n); //expecation -> 1 2 3 4....n-2 n-1 n
}

public static void printIncreasing(int n){
1   if(n == 0) {
        return;
    }

    //faith -> 1 2 3 4...n-2 n-1
2   printIncreasing(n-1);

3   System.out.println(n);
}
```



```java
public static void main(String[] args) throws Exception {
    // write your code here
1   Scanner scn = new Scanner(System.in);
2   int n = scn.nextInt();

3   printIncreasing(n); //expecation -> 1 2 3 4....n-2 n-1 n
}

public static void printIncreasing(int n){
1   if(n == 0) {
        return;
    }

    //faith -> 1 2 3 4...n-2 n-1
2   printIncreasing(n-1);

3   System.out.println(n);
}
```





expectation pdi(4) → 4 3 2 1 1 2 3 4

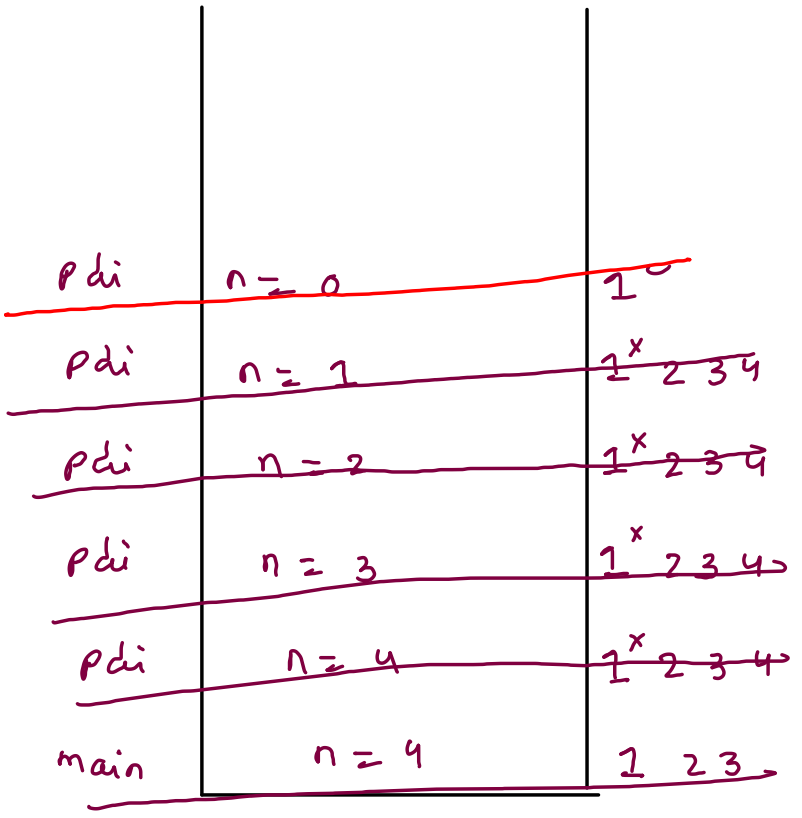
jaith pdi(3) → 3 2 1 1 2 3

pdi(4) → syso(4) + pdi(3) + syso(4)

4 3 2 1 1 2 3 4 4 3 2 1 1 2 3 4

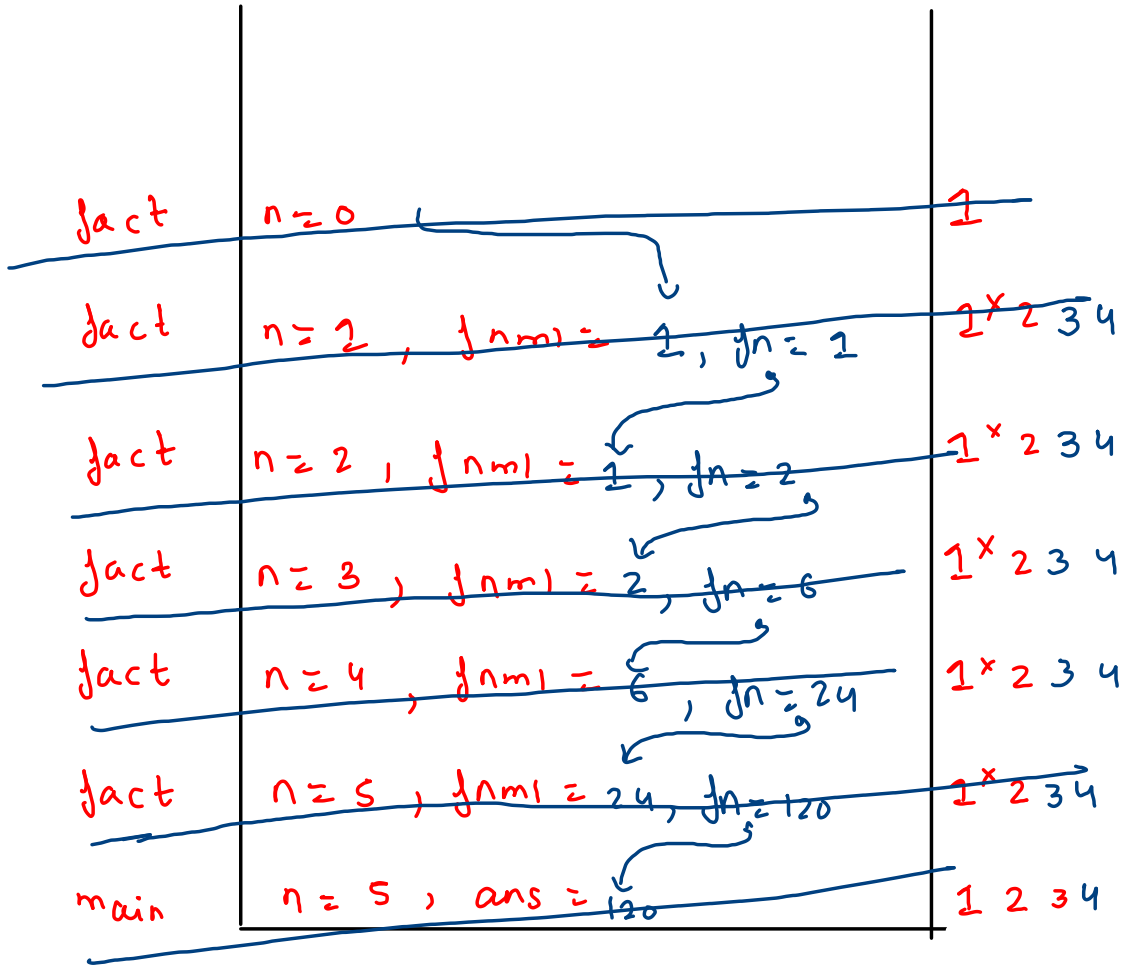
```java
public static void main(String[] args) throws Exception {
    // write your code here
1   Scanner scn = new Scanner(System.in);

2   int n = scn.nextInt();
3   pdi(n);
}

public static void pdi(int n){
1   if(n == 0) {
        return;
    }

2   System.out.println(n);
3   pdi(n-1); //faith
4   System.out.println(n);
```

4 3 2 1 1 2 3 4



```java
public static void main(String[] args) throws Exception {
    // write your code here
1   Scanner scn = new Scanner(System.in);
2   int n = scn.nextInt();
3   int ans = factorial(n);
4   System.out.println(ans);
}

public static int factorial(int n){
1   if(n == 0) {
        //0! = 1
        return 1;
    }

2   int fnm1 = factorial(n-1); //factorial of n-1
3   int fn = n*fnm1; //factorial of n

4   return fn;
}
```

120



power_linear

expectation: pow(x, n) → x^n

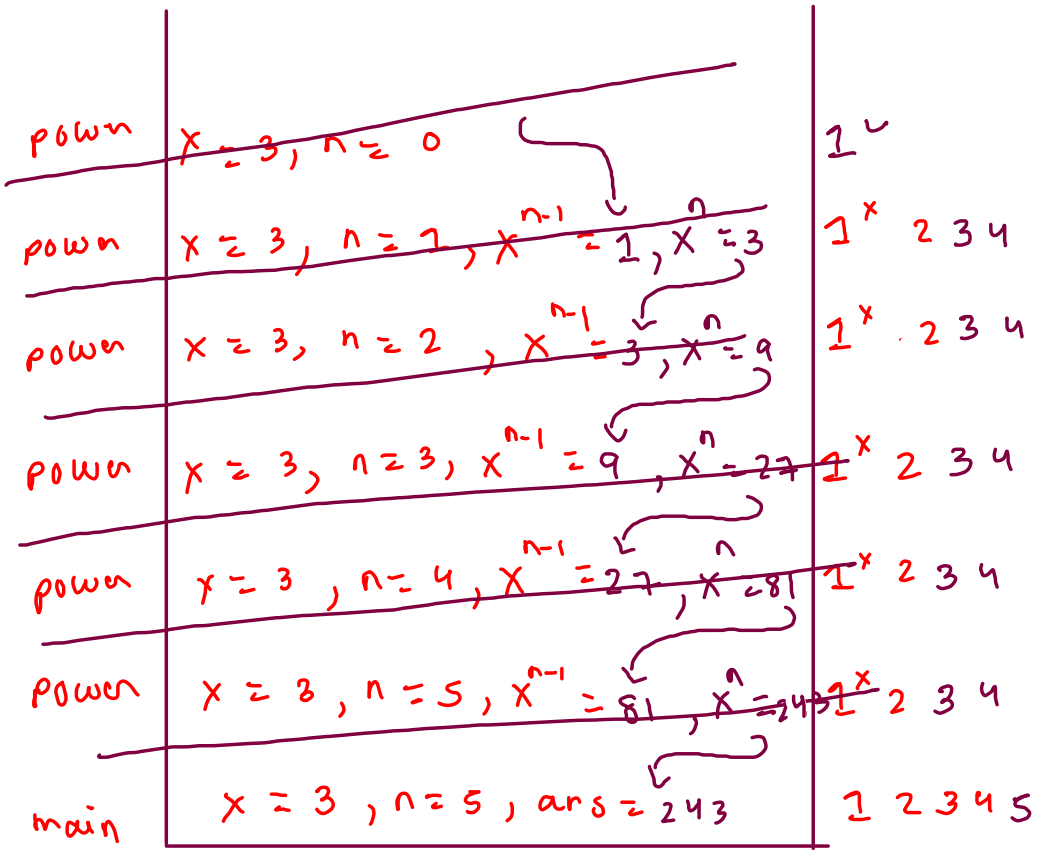
faith: pow(x, n-1) → x^{n-1}

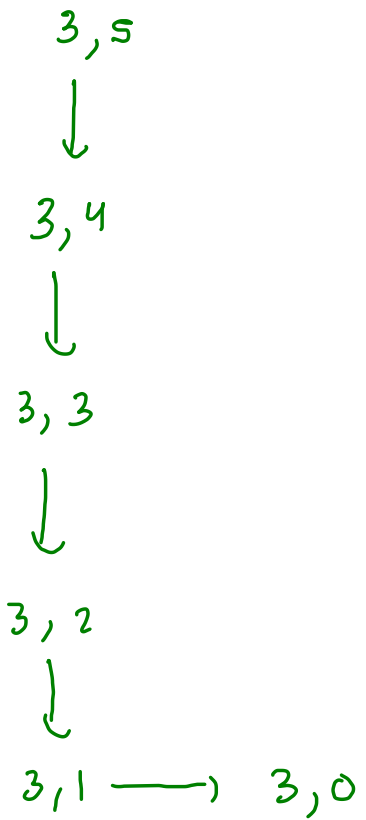
pow(x, n) = x * pow(x, n-1)

x^n = x * x^{n-1}

```java
public static void main(String[] args) throws Exception {
    // write your code here
1   Scanner scn = new Scanner(System.in);
2   int x = scn.nextInt();
3   int n = scn.nextInt();

4   int ans = power(x,n);
5   System.out.println(ans);
}

public static int power(int x, int n){
1   if(n == 0) {
        return 1;
    }

2   int xpnm1 = power(x,n-1); //x raised to the power n-1
3   int xpn = x * xpnm1; //x raised to the power n

4   return xpn;
}
```

243



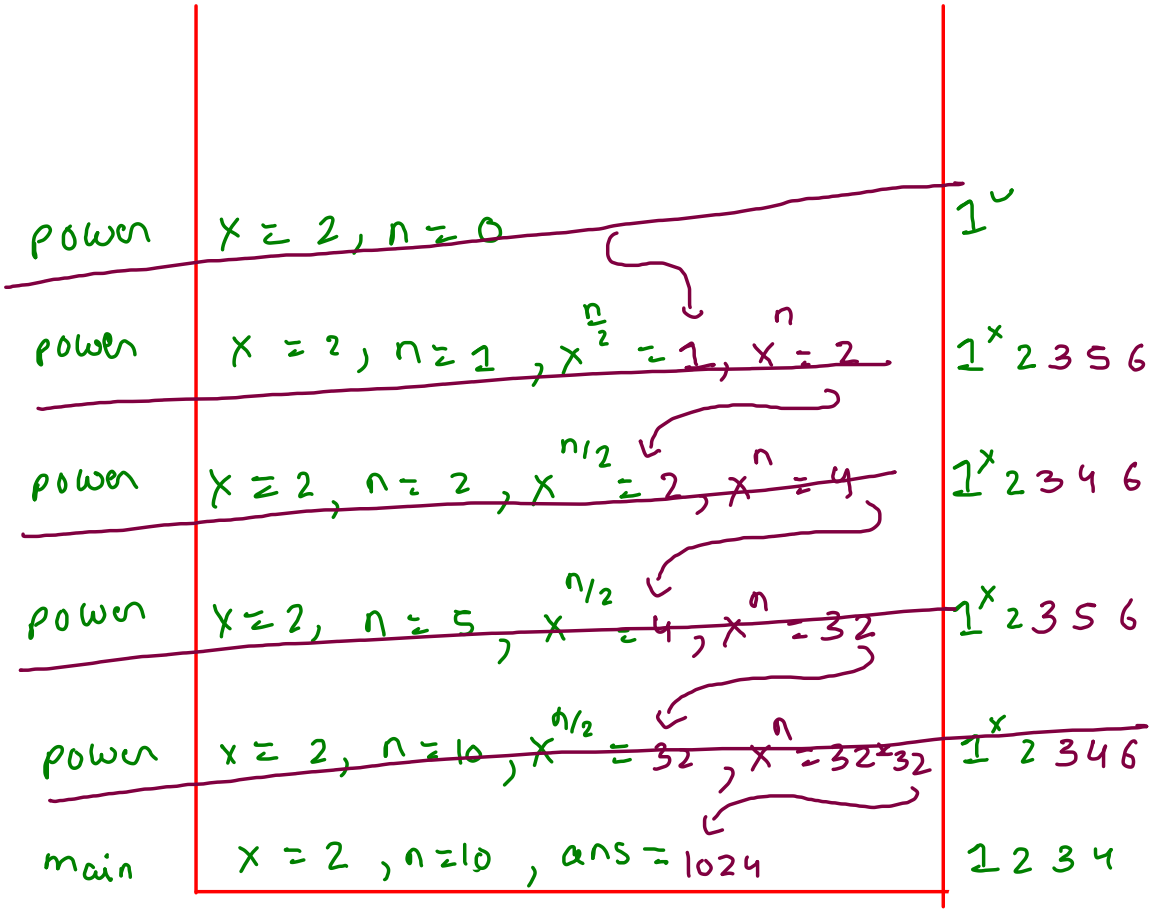


x^n → n or

$$pow(x,n) = \begin{cases} x^{\frac{n}{2}} * x^{\frac{n}{2}} & n \text{ is even} \\ x^{\frac{n}{2}} * x^{\frac{n}{2}} * x & n \text{ is odd} \end{cases}$$

```java
public static void main(String[] args) throws Exception {
    // write your code here
1   Scanner scn = new Scanner(System.in);
2   int x  = scn.nextInt();
3   int n = scn.nextInt();

4   int ans = power(x,n);
5   System.out.println(ans);
}

public static int power(int x, int n){
1   if(n == 0) {
        return 1;
    }

2   int xpnb2 = power(x,n/2); //x raised to the power n/2

3   int xpn = 1;

    if(n % 2 == 0) {
4       xpn = xpnb2 * xpnb2;
    }
    else {
5       xpn = xpnb2 * xpnb2 * x;
    }

6   return xpn;
}
```

1024



```java
public static int power(int x, int n){
    if(n == 0) {
        return 1;
    }

    int xpnm1 = power(x,n-1); //x raised to the power n-1
    int xpn = x * xpnm1; //x raised to the power n

    return xpn;
}
```

2,10 → 2,9 → 2,8 → 2,7 → 2,6 → 2,5 → 2,4 → 2,3 → 2,2 → 2,1 → 2,0

opr : n

n → n-1 → n-2 → ⋮ → 0

n, n-1, n-2, ------ 0

k → total terms

a = n, d = -1

$t_t = a + (k-1)d$

$0 = n + (k-1)(-1)$

$k-1 = n$

$k = n + 1$

$k \propto n$

```java
public static int power(int x, int n){
    if(n == 0) {
        return 1;
    }

    int xpnb2 = power(x,n/2); //x raised to the power n/2

    int xpn = 1;

    if(n % 2 == 0) {
        xpn = xpnb2 * xpnb2;
    }
    else {
        xpn = xpnb2 * xpnb2 * x;
    }

    return xpn;
}
```

2,10 → 2,5 → 2,2 → 2,1 → 2,0

n → n/2 → n/4 → ⋮ → 0

log n

opr = log n

$n, \frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \ldots\ldots 1$

$t_t = ar^{(k-1)}$

a = n

$r = \frac{1}{2}$

$1 = n\left(\frac{1}{2}\right)^{k-1}$

$2^{k-1} = n$

$k-1 = \log_2 n$

$k = \log_2 n + 1$

$k \propto \log_2 n$

```
fun ( ) {

    for (c1) {

        for (c2) {

            [for (c3) {
                break;
            }

        }

}
```