$5! = 1 \times 2 \times 3 \times 4 \times 5$

Iterative

$5! = 4! \times 5$

Recursion

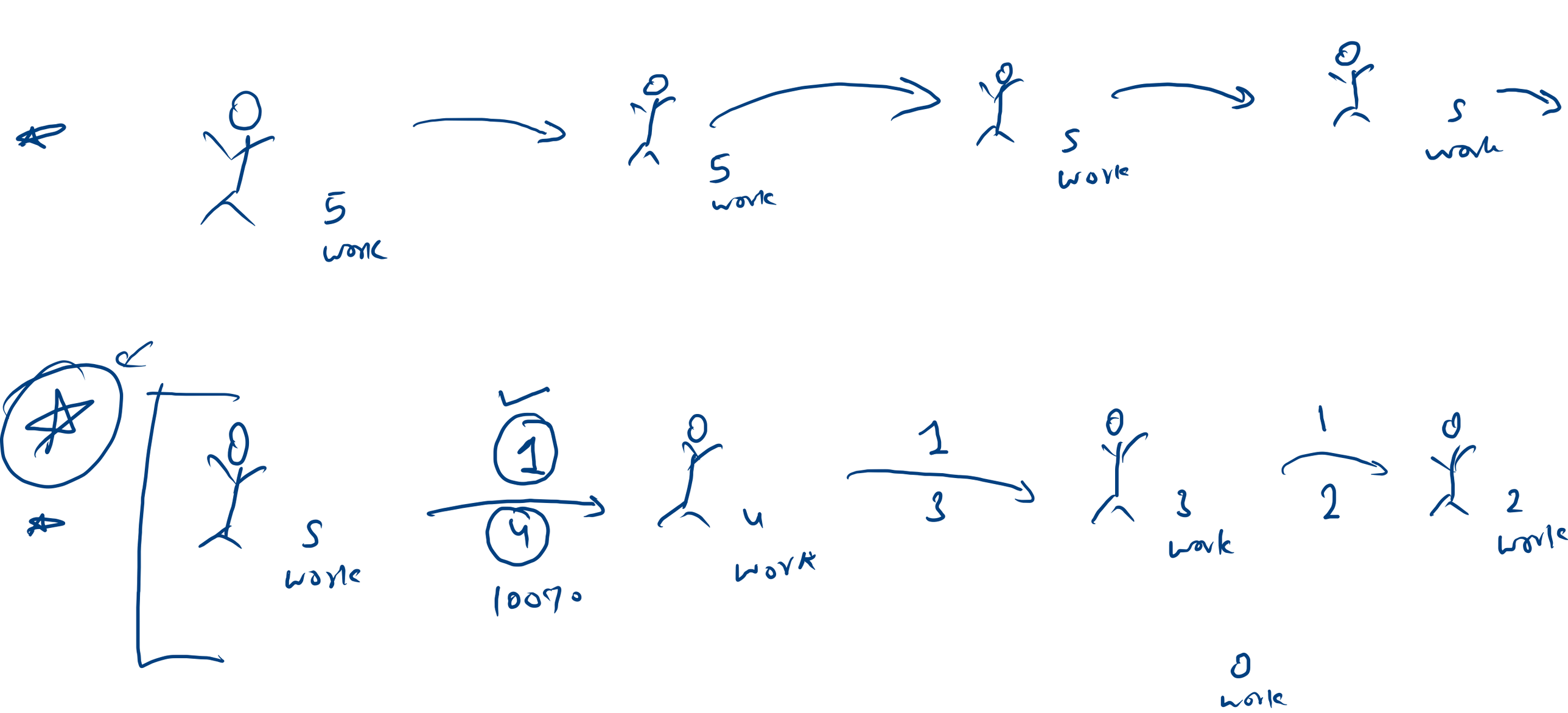
$5!$

expectation

$4!$

faith

100%


5
work
5
work
5
work
5
work
1
100%
4
work
1
3
3
work
1
2
2
work
0
work

h → 5

5
4
3
2
1

recursion → higher level

lower level

expectation

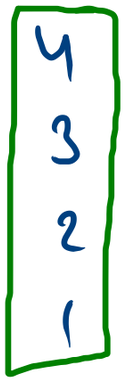
pd(5)

5

faith

faith → 100% ←

pd(4)



exp

```
[ print    work
  faith(4) faith
```

n → 5

5
4
3
2
1

recursion

higher level

lower level

explanation

pd(n)

n

n-1
n-2
n-3
:
1

faith

faith 100%

pd(n-1)

n-1
n-2
n-3
n-4
:
1

print(n)

faith(n-1)

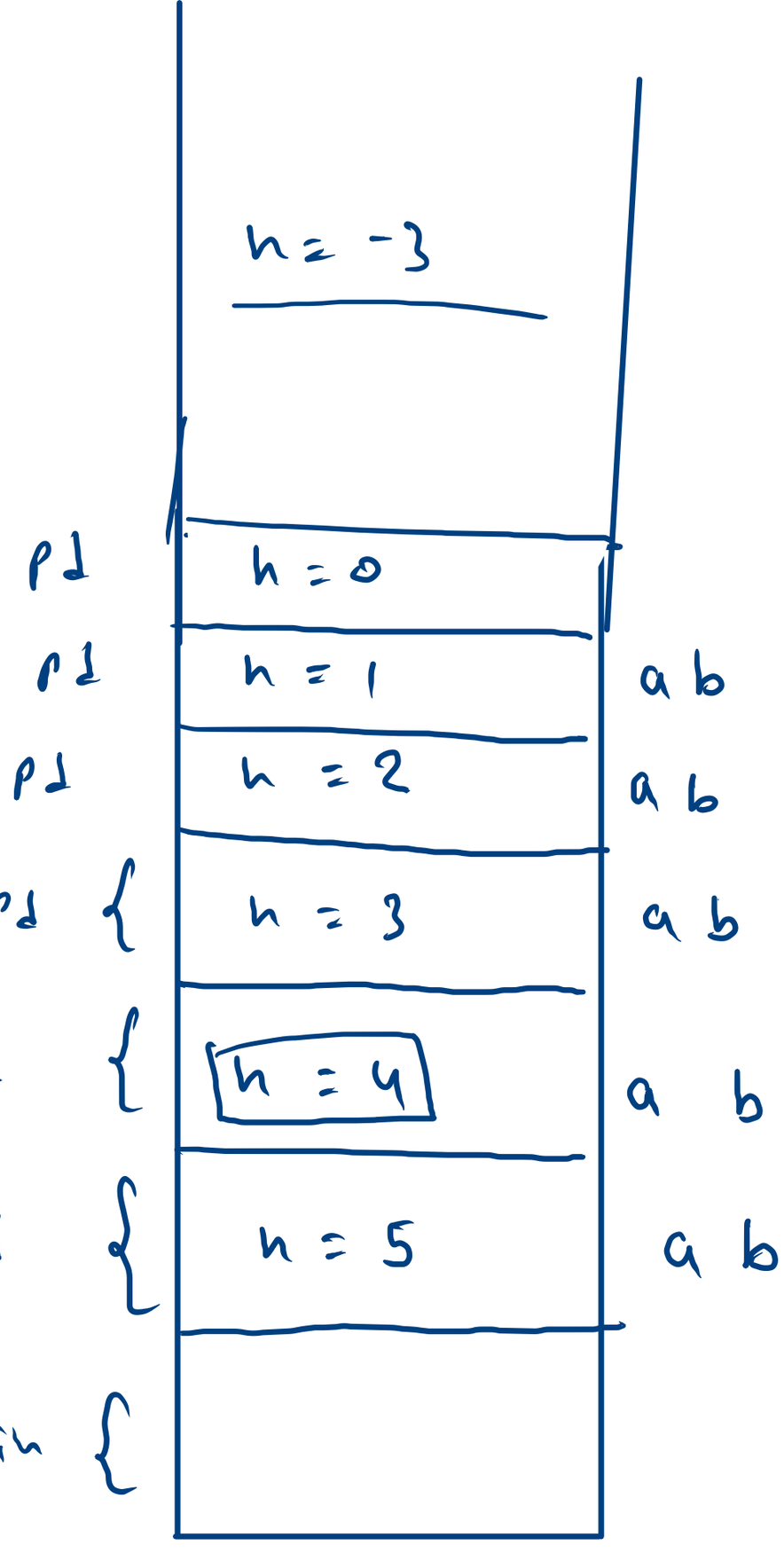
n = 5

-1
-2
-3

```
public static void main(String[] args) throws Exception {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    printDecreasing(n);
}

public static void printDecreasing(int n){
    if(n==0) return;
    System.out.println(n);   a ←
    printDecreasing(n-1);    b
}
```

5
4
3
2
1 ←

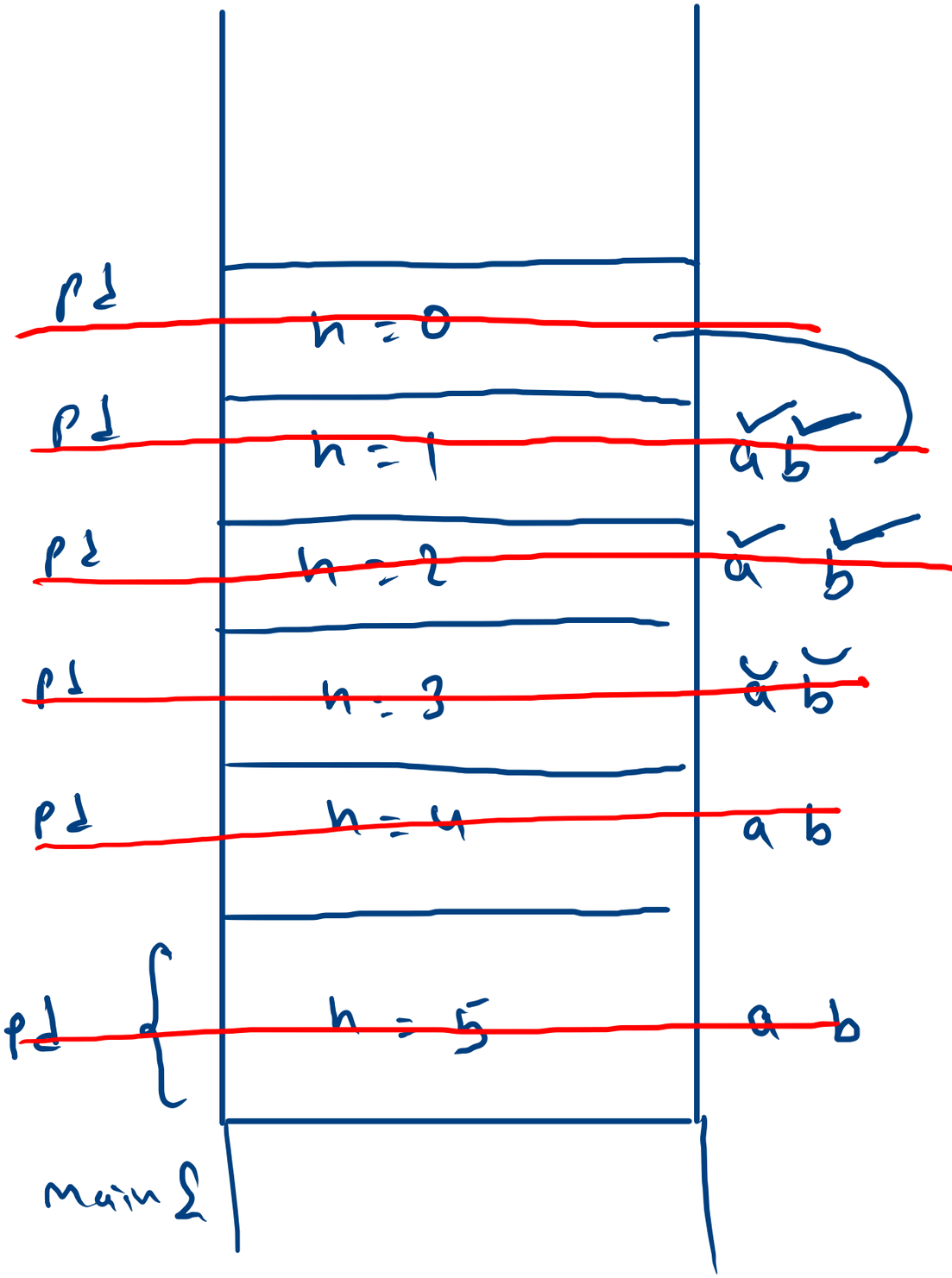


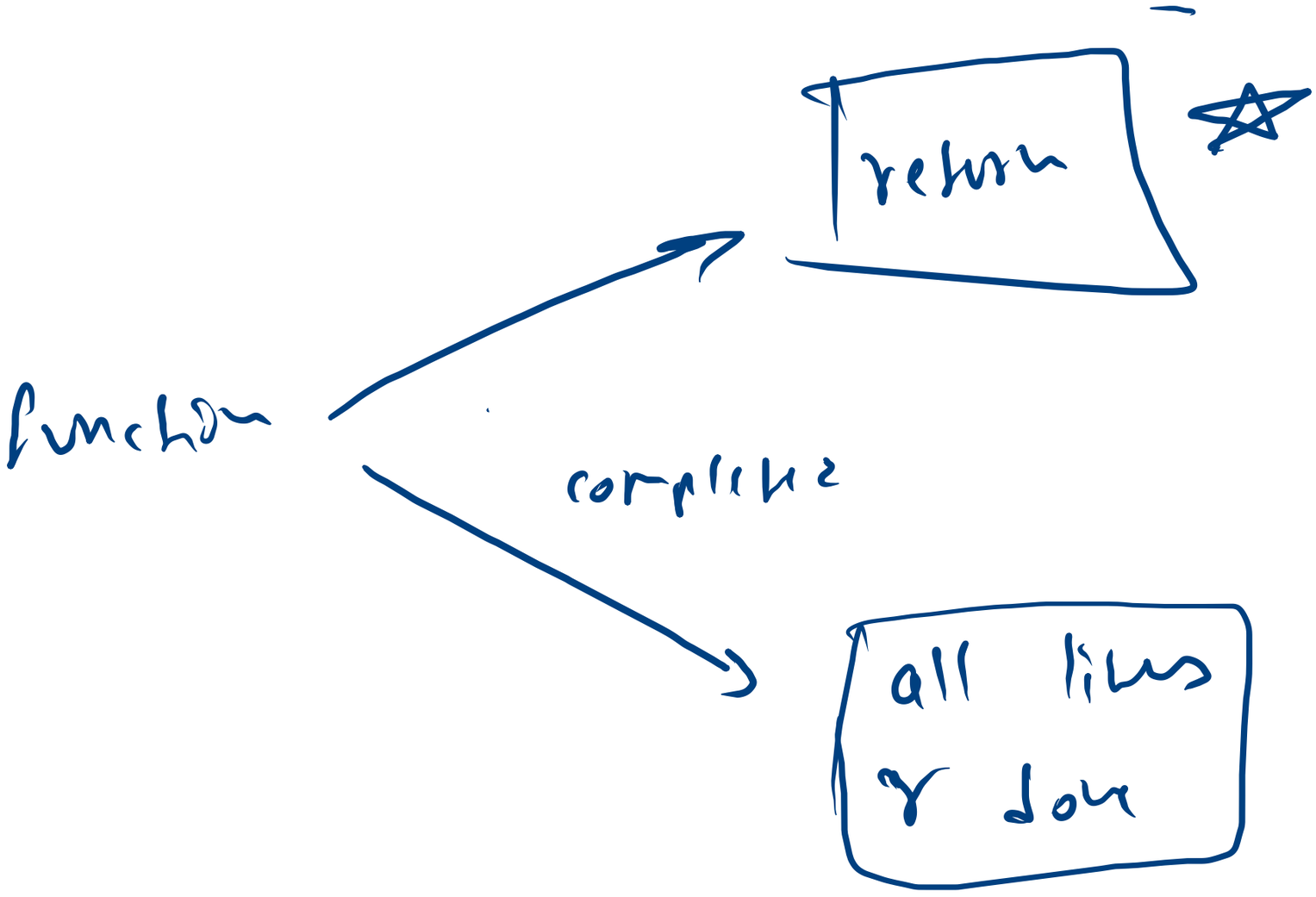
```java
public static void printDecreasing(int n){
    if(n == 0){
        return;
    }

    System.out.println(n);   a
    printDecreasing(n-1);    b
}
```

base condition

5
4
3
2
1




return
function
complete
all lines
return

n = 5

1
2
3
4
5

expectation

PI(5)

1
2] faith
3
4

5] work

faith

PI(4)

1
2
3
4

expec
PI(5) [faith(4)
print(5)

n = 3

PI(n)

PI(n-1)

PI(n-1)
print(n)

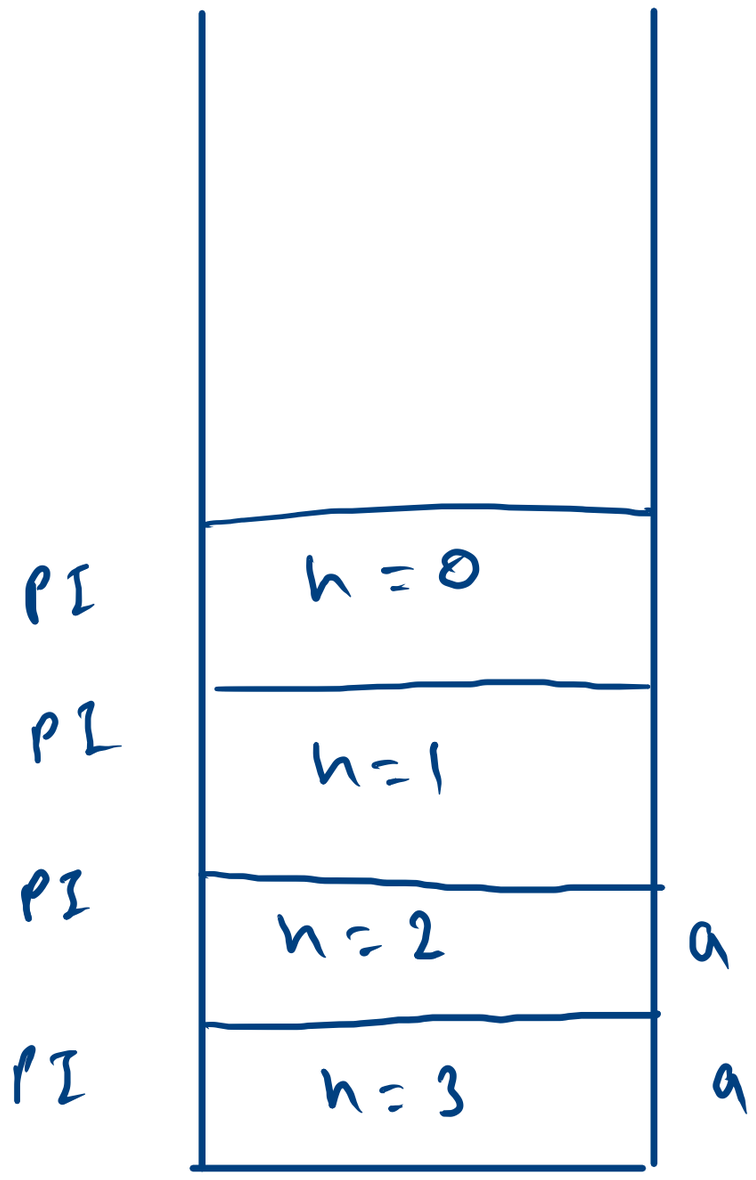
1
2
3

n = 3

```
public static void printIncreasing(int n){

    printIncreasing(n-1);    a
    System.out.println(n);   b

}
```

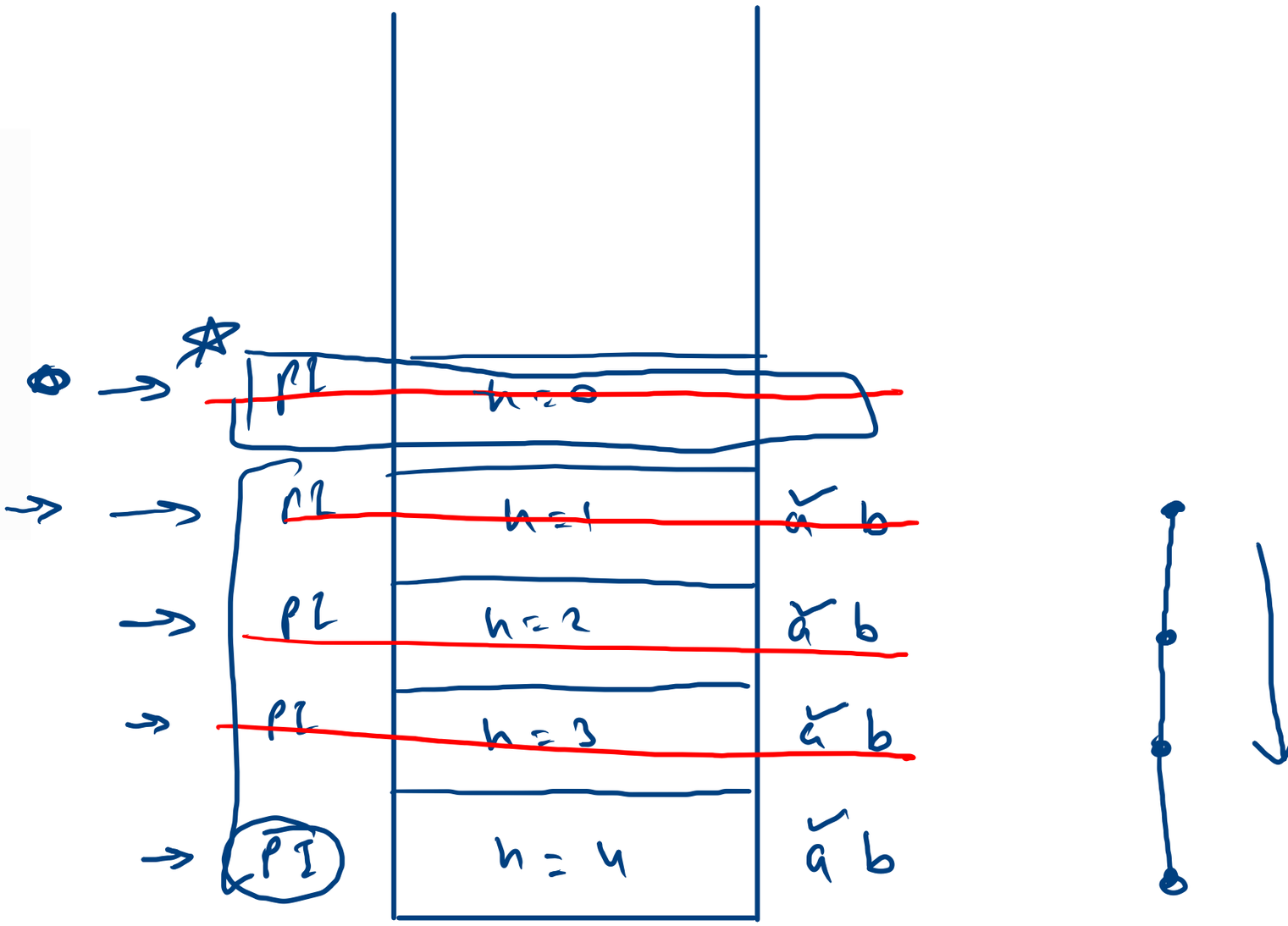


```java
public static void printIncreasing(int n){
    if(n == 0){
        return;
    }

    printIncreasing(n-1);
    System.out.println(n);

}
```

1
2
3
4

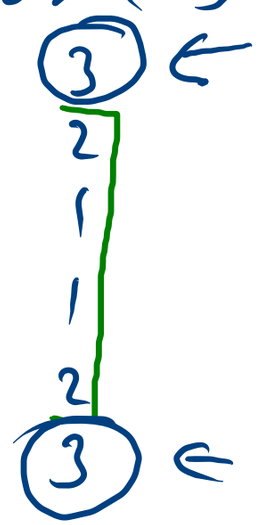


n = 3

3
2
1
1
2
3

execution

PDI (3)



Pdi (n)

Pdi (2)

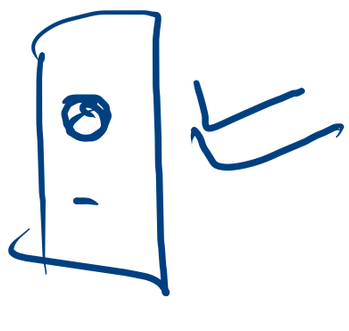
Pdi (n-1)

Pdi (3)

expect
Print (3)
Pdi (2)
Print (3)

Print
Pdi (n-1)
Print

3 2 1
3 2 1
2 1 1
1 1 2
1 1
2
3

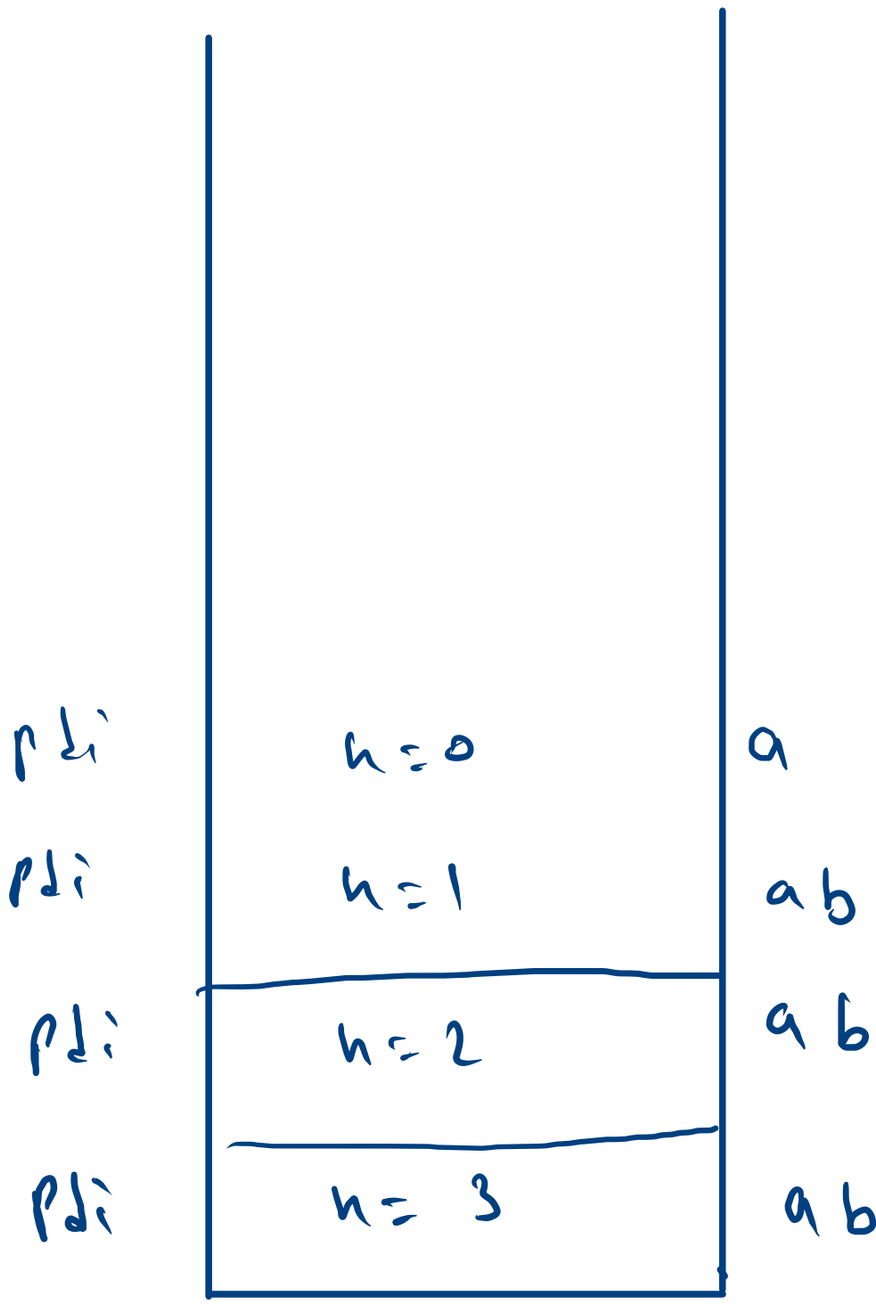


```
public static void pdi(int n){

    System.out.println(n);
    pdi(n-1);
    System.out.println(n);
}
```

a
b
c

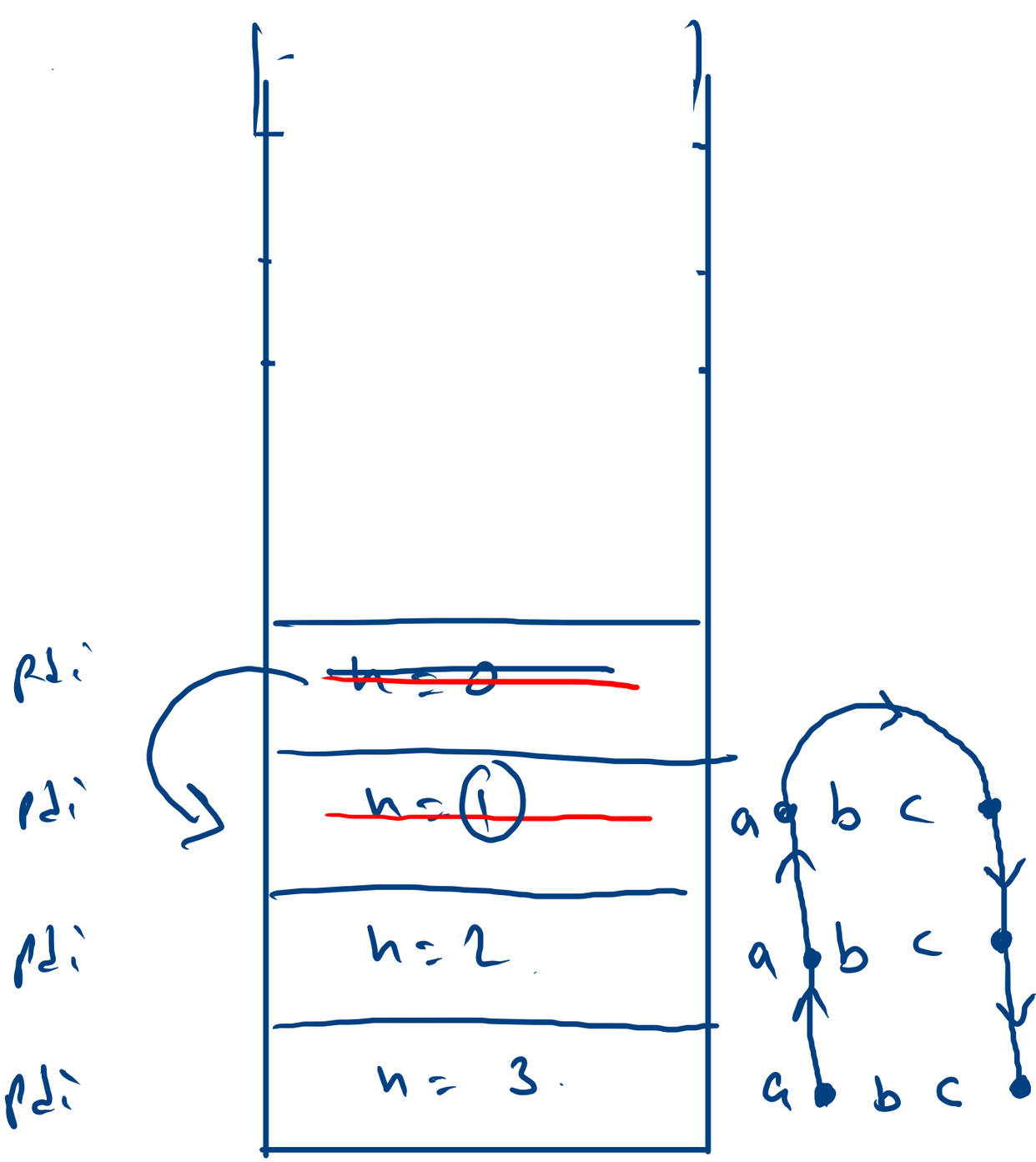
3
2
1



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

    System.out.println(n);
    pdi(n-1);
    System.out.println(n);
}
```

3
2
1
1
2
3



n = 3

= 3 x 2 x 1

= 6

n = 0

= 1

expectation

f(4)

4 x 3 x 2 x 1

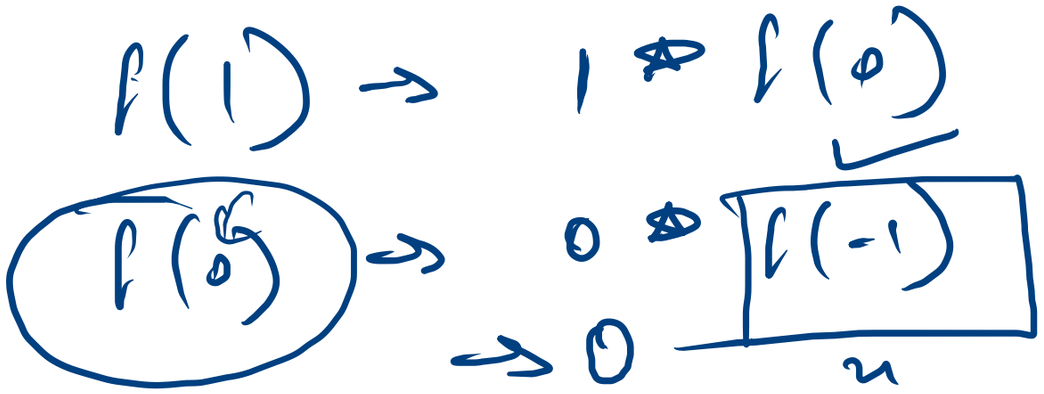
faith

faith

f(3)

3 x 2 x 1

$f(n) = n * f(n-1)$



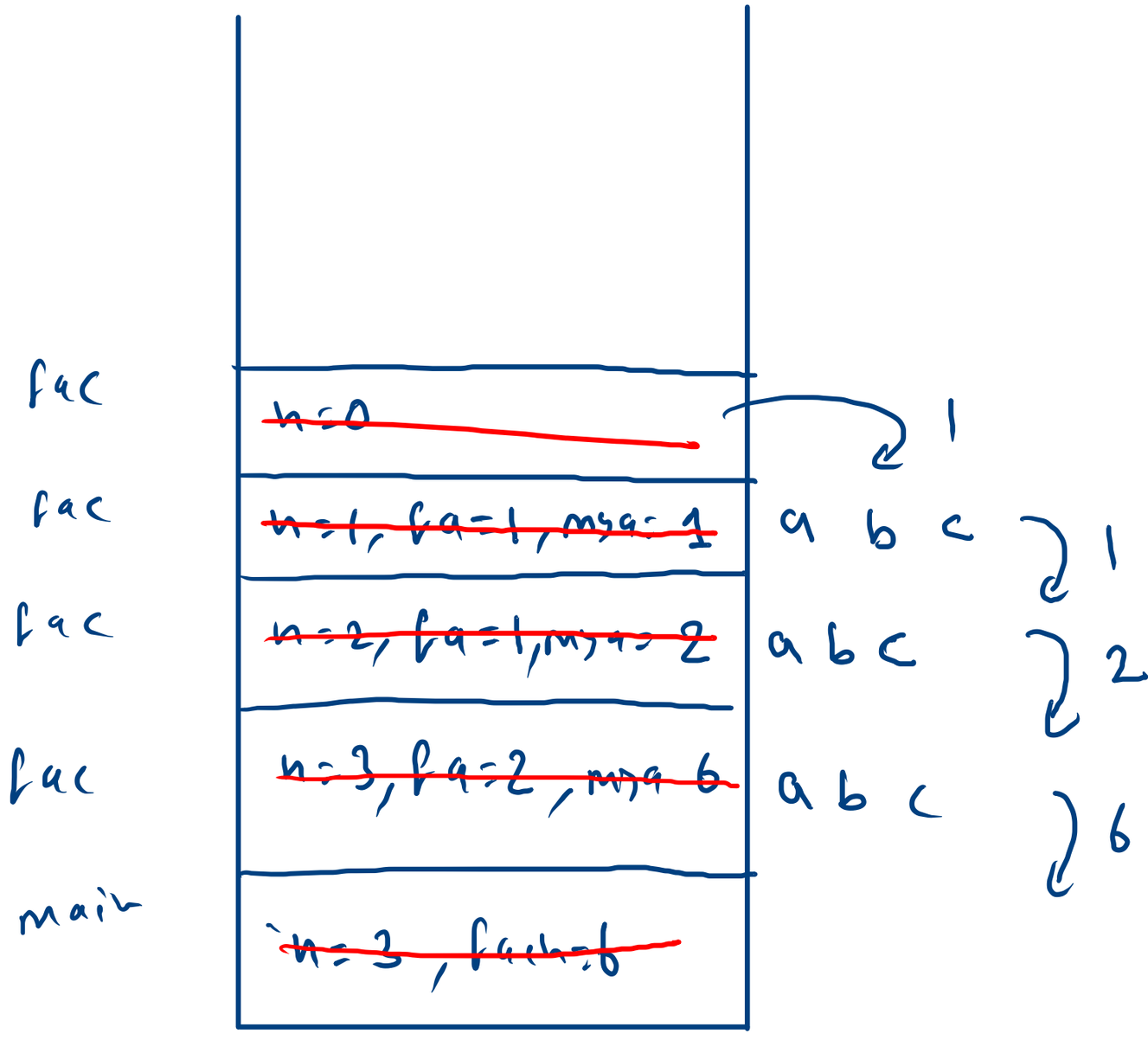

```java
public static int factorial(int n){
    if(n==0) return 1;
    int fa = factorial(n-1);
    int mya = n*fa;
    return mya;
}
```

n=3

```java
public static void main(String[] args) throws Exception {
  Scanner scn = new Scanner(System.in);
  int n = scn.nextInt();
  int facn = factorial(n);
  System.out.println(facn);
}

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

  int fa = factorial(n - 1);
  int mya = n * fa;
  return mya;
}
```



$n = 2$

$h = 5$

$n^n \quad 2^5 = 2 \times 2 \times 2 \times 2 \times 2$

32

expectation

2^5

$2 \times \boxed{2 \times 2 \times 2 \times 2}$

2^4

faith

2^4

$2 \times 2 \times 2 \times 2$

$n -$

$n = 0$

$n^0 \to 1$

$f(n, n) = n * f(n, n-1)$

base $\quad n^0 \to 1$

$n^n \rightarrow n * n^{n-1}$

$n \rightarrow 2$

$n \rightarrow 5$

$2^5 = 2 \times 2 \times 2 \times 2 \times 2$

2×2^4

$2x \qquad 2^4$

$n = 2,\ n = 4$

even $n = 2$

$n = 6$

$2^6 = 2^3 \times 2^3 \rightarrow 2^6$

$n^n = n^{n/2} * n^{n/2}$

odd $n = 2$

$n = 5$

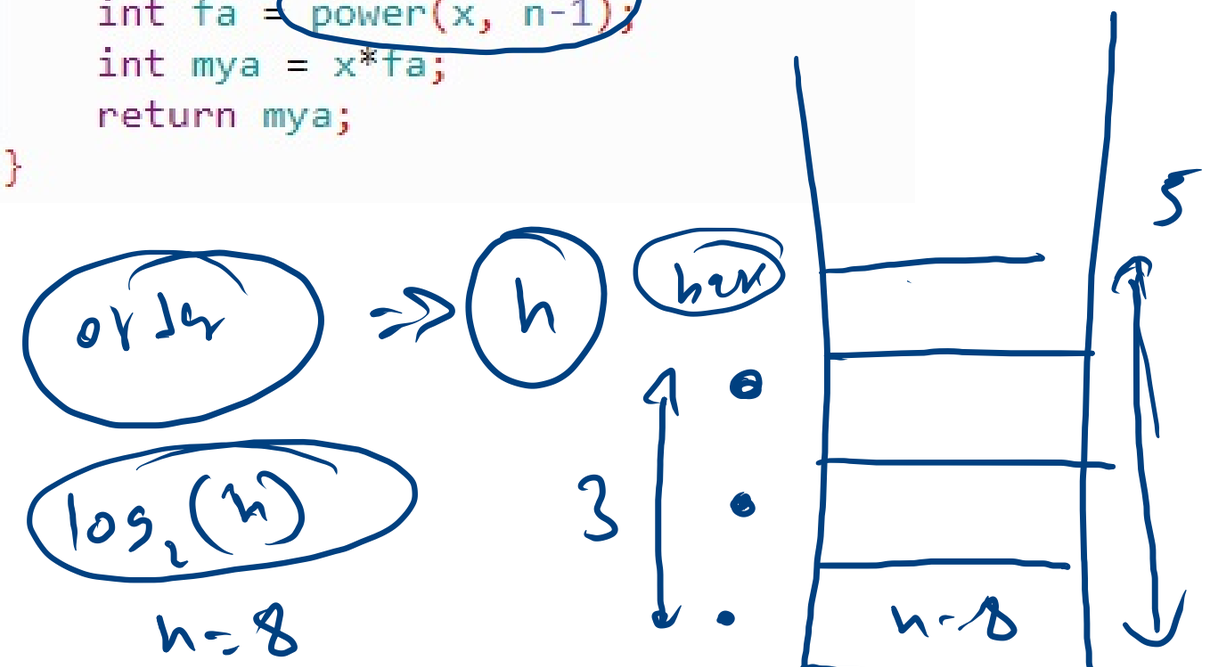
$2^5 = 2^2 \times 2^2 \times 2$

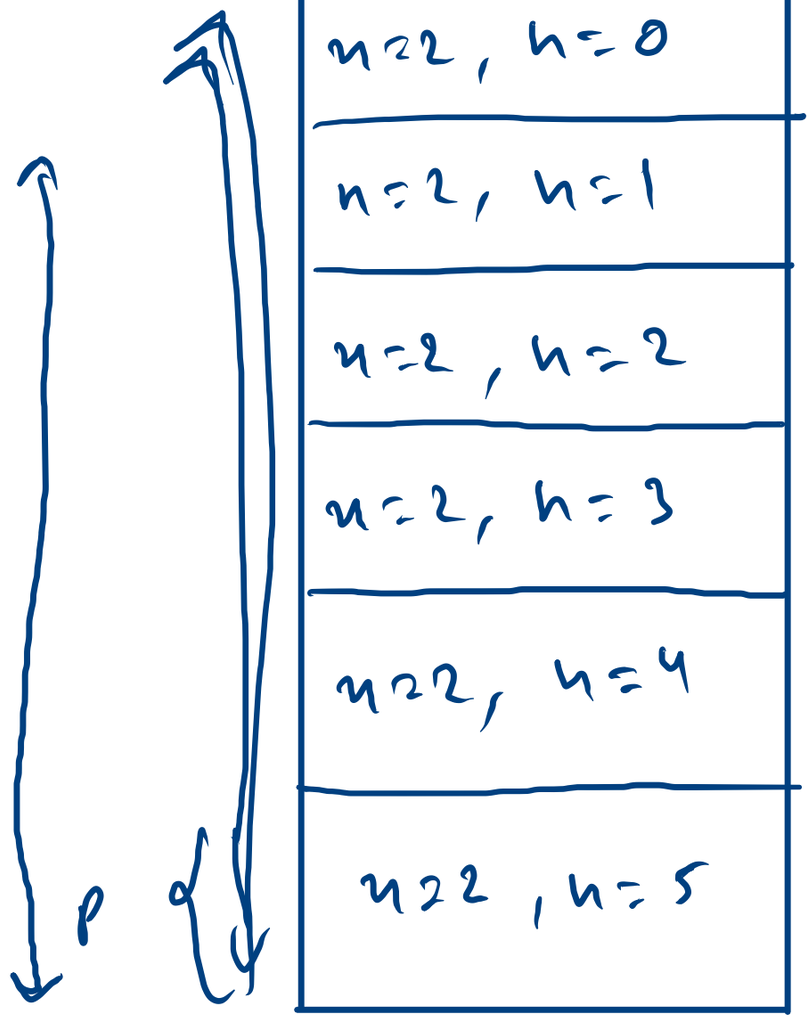
x = 2

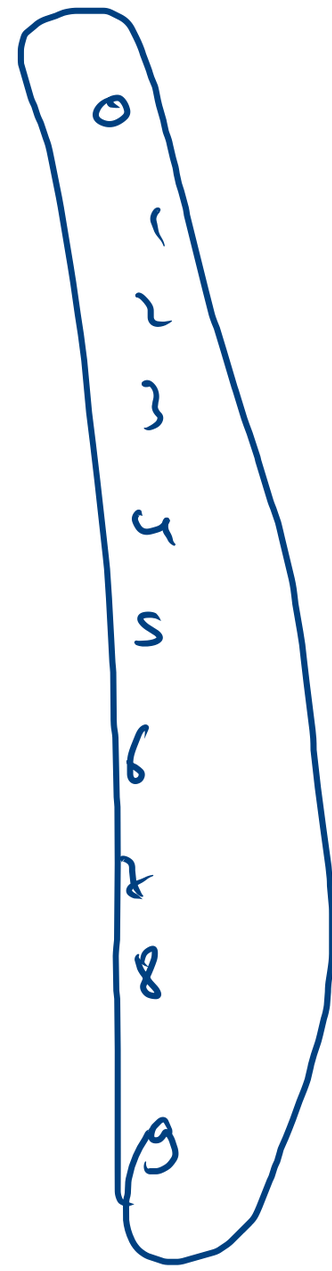
x2 ~~8~~ 9

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

    int fa = power(x, n-1);
    int mya = x*fa;
    return mya;
}
```







even

$n = 2$

$n = 8$

$2^6 \Rightarrow 2^4 \times 2^4$

$n^4 =$ ($n^{4/2}$ × $n^{4/2}$)

$f_4 \times f_4$

odd

$n = 2$

$n = 9$

$2^9 = 2^4 \times 2^4 \times 2$

$n^n = n^{n/2} \times n^{n/2} \times n$

$f_4 \times f_4 \times n$

exponentiation

n^n $\quad$ 2^5

pair

$a = b(n, n/2)$

$n = 2$

$n = 8$

$\log_2(n)$

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

    int fa = power(x, n/2);

    if( n %2 == 0 ){
        int ans = fa * fa;
        return ans;
    }else{
        int ans = fa * fa * x;
        return ans;
    }
}
```

a

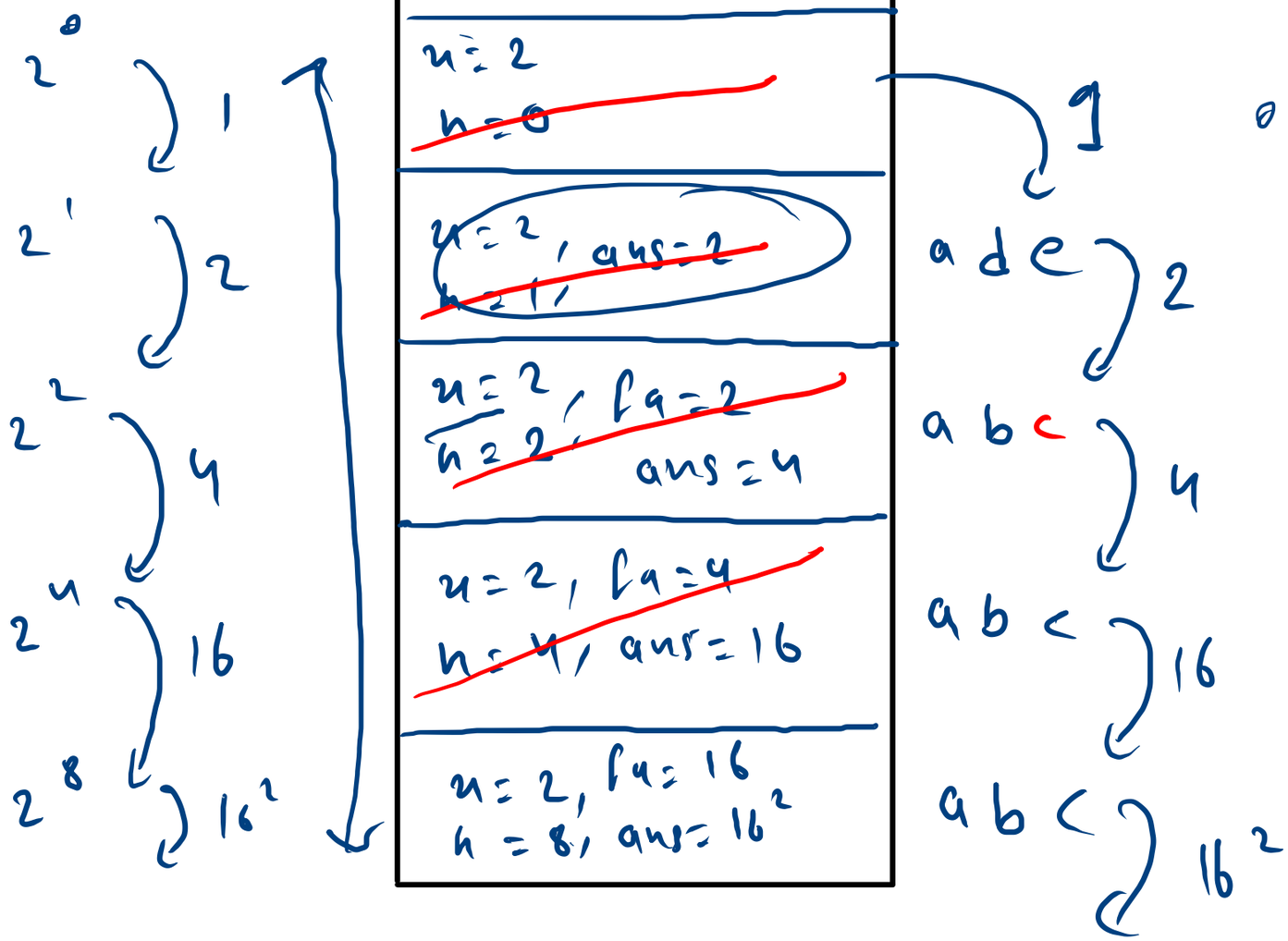
b

c

d

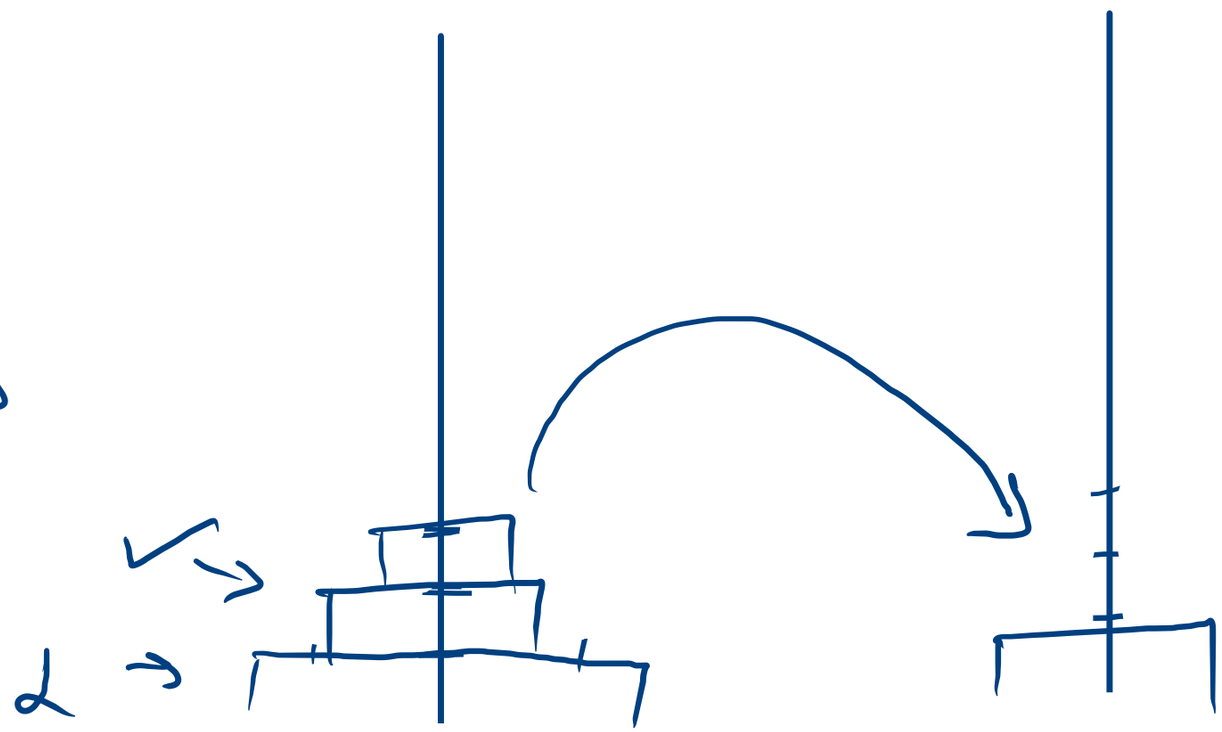
e

ans = 16×16



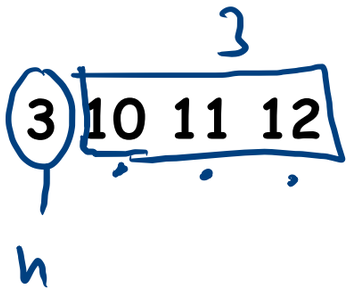
- 3.3.1 move 1 disk at a time.
- 3.3.2 never place a smaller disk under a larger disk.
- 3.3.3 you can only move a disk at the top.

A→B

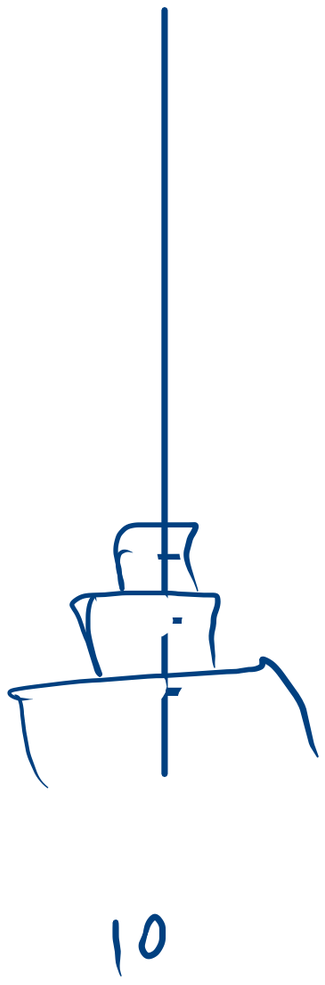


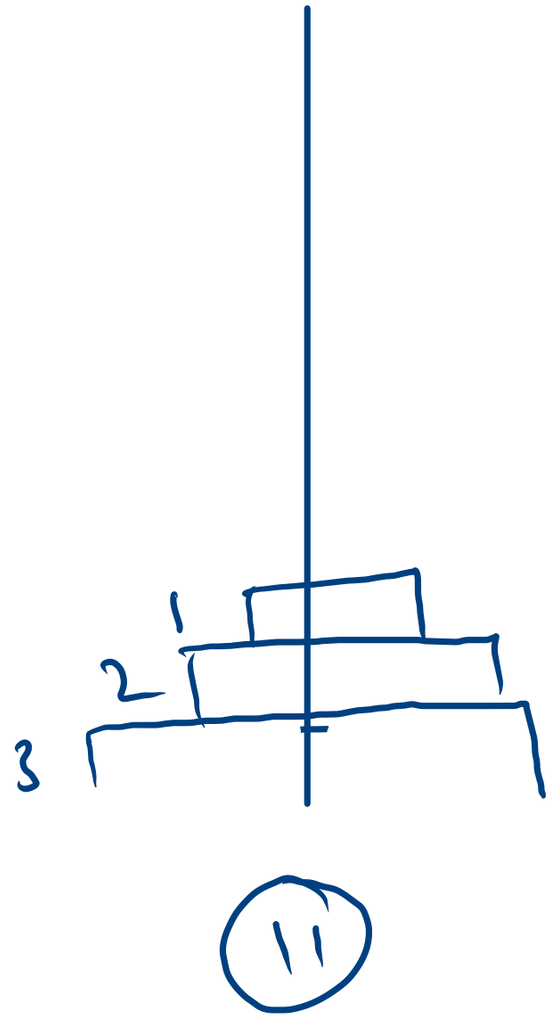
A

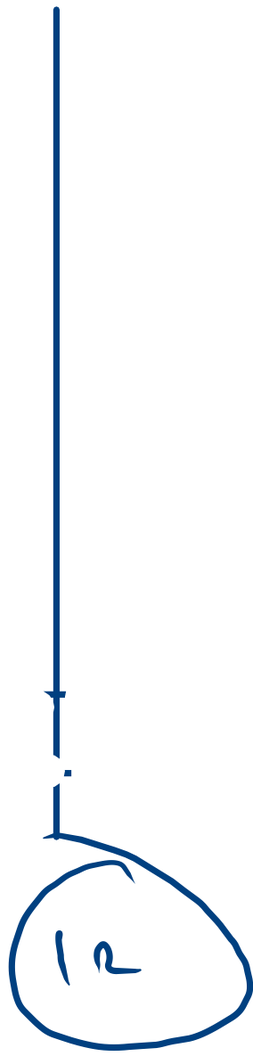
C

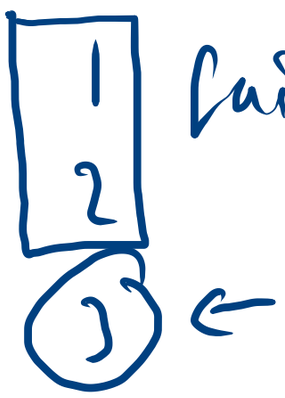


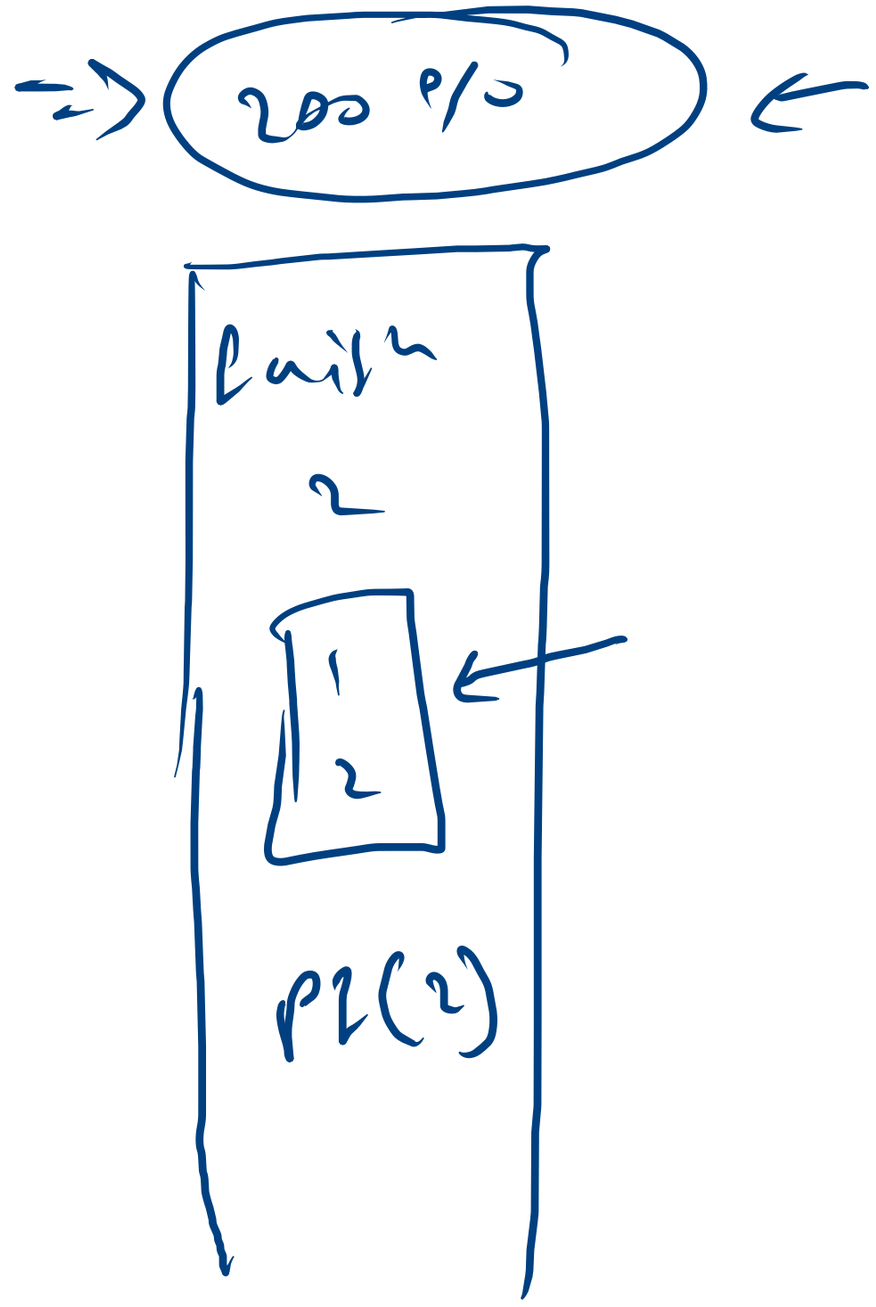
1[10 -> 11]
2[10 -> 12]
1[11 -> 12]
3[10 -> 11]
1[12 -> 10]
2[12 -> 11]
1[10 -> 11]





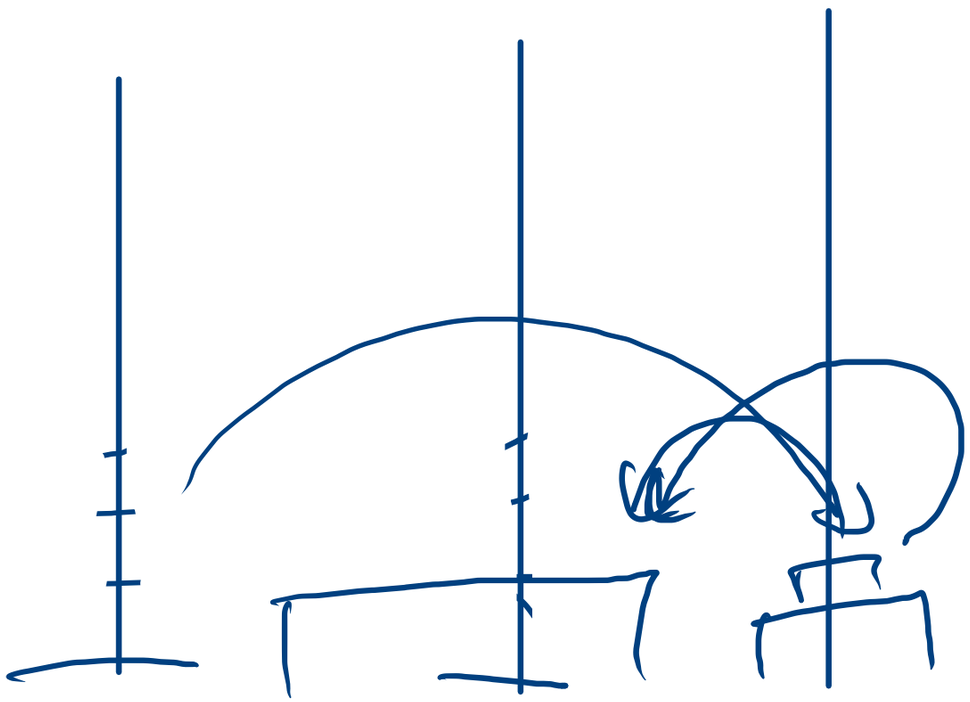






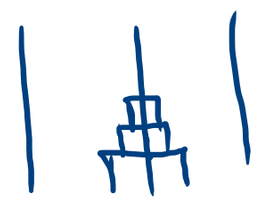
0 → recur

1 →



expectation

3A → B



help

faith 2007.

2A → C, B

⇒ print 3A → B

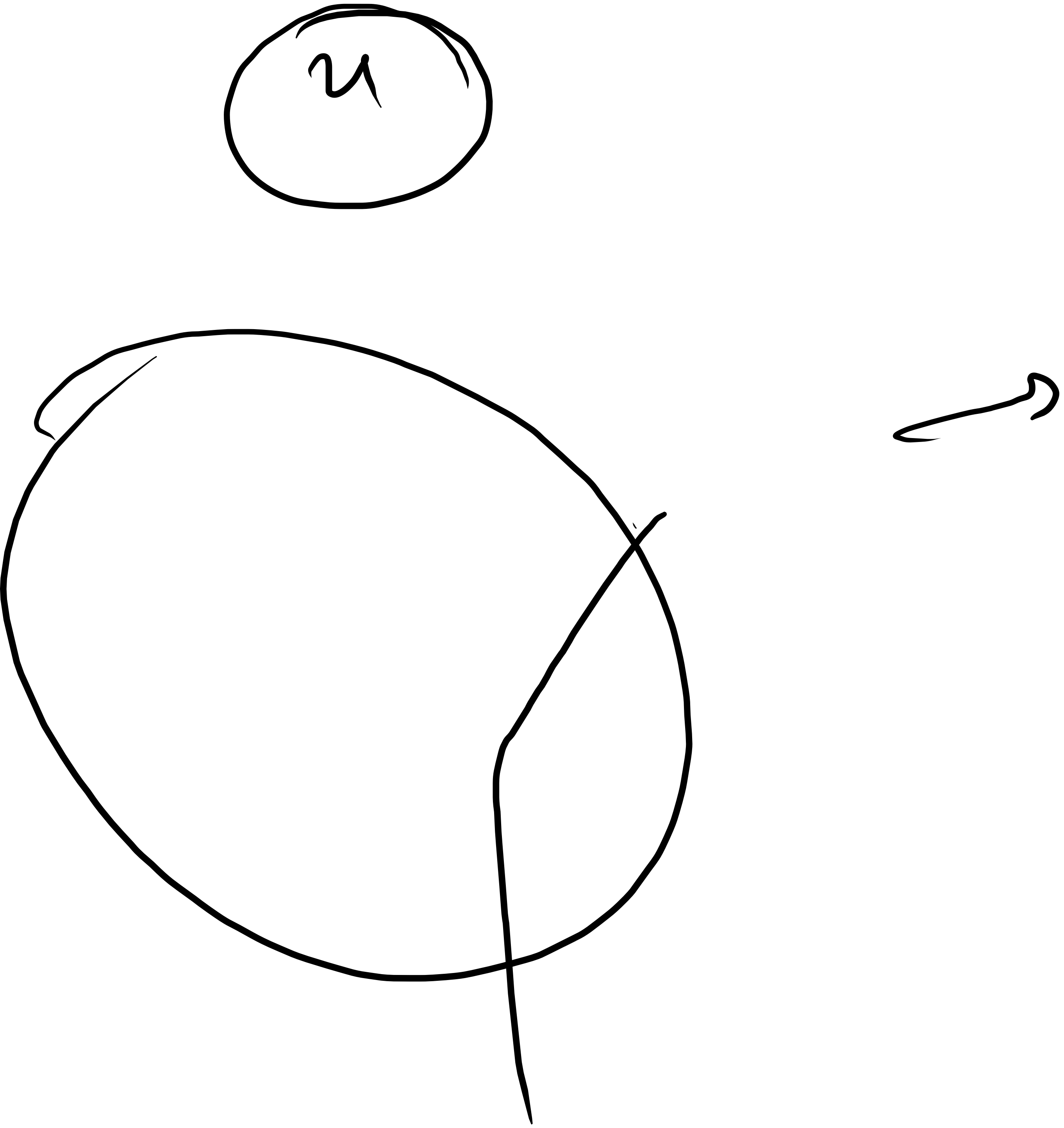
2C → B, A

nA → B

(n-1) A → C

print [n] A → B

(n-1) C → B



cain(- < u)

(- > u) α

```java
public static void toh(int n, int A, int B, int C){
    if(n==0){
        return;
    }
    toh(n-1, A, C, B);
    System.out.println(n+"["+A+" -> "+B+"]");
    toh(n-1, C, B, A);
}
```

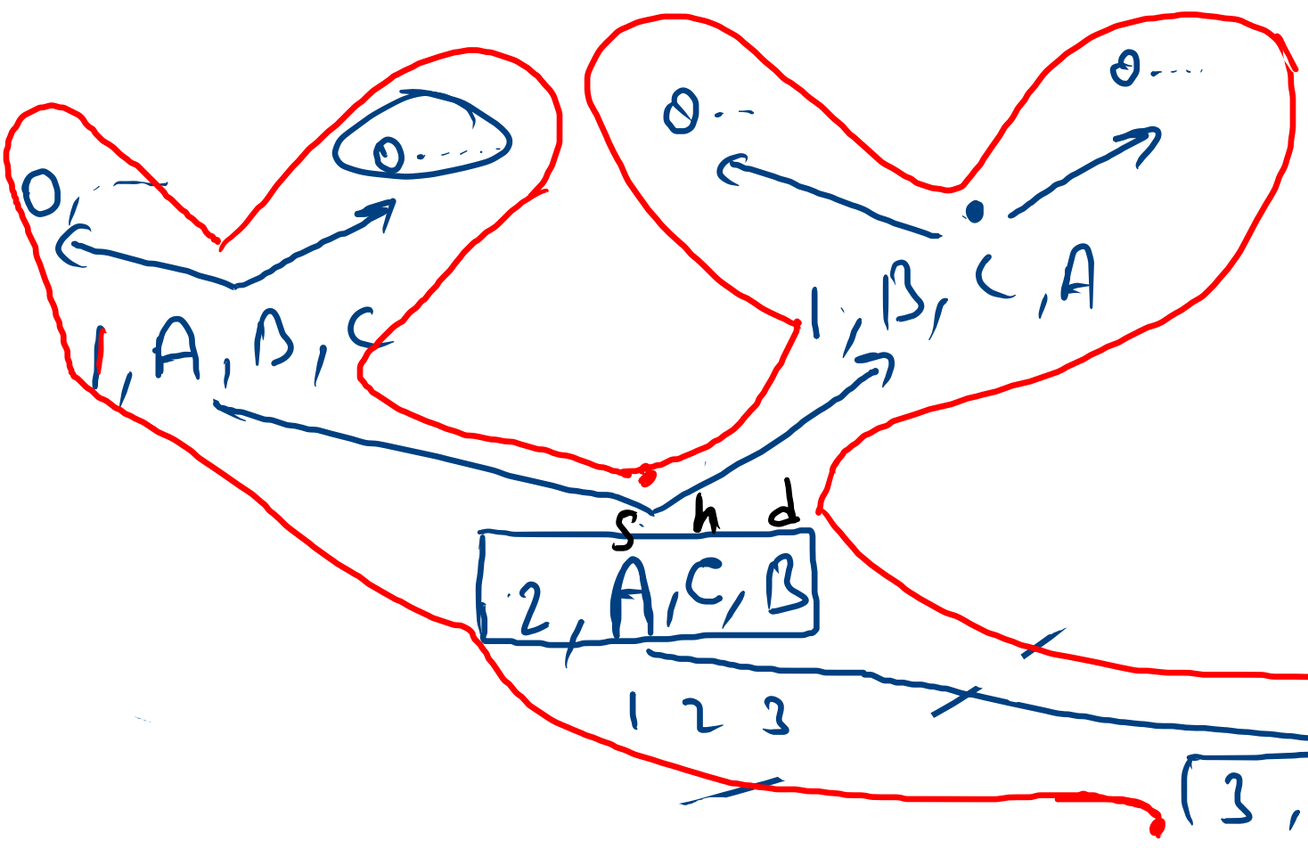
src des help

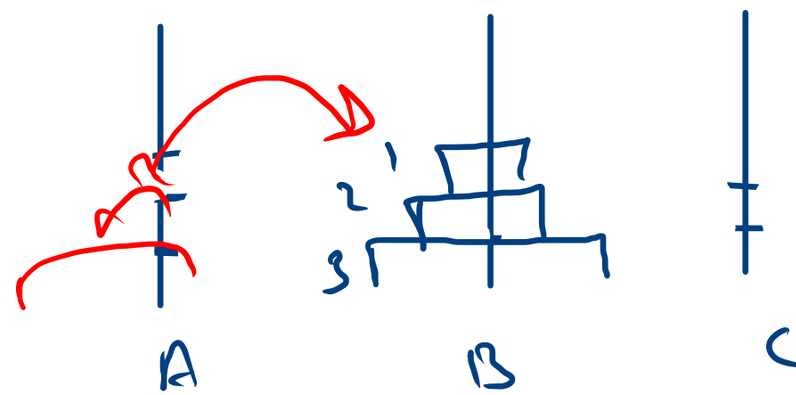
src help des

left

right

help des src





1 A→B
2 C→B
1 C→A
3 A→B
1 B→C
2 A→C
1 A→B