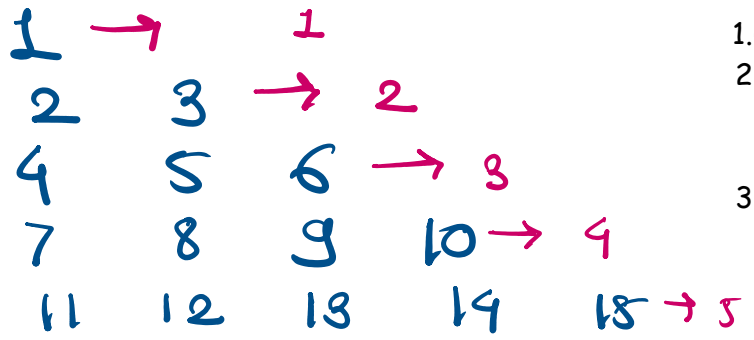
1. Observation
2. Break the pattern into smaller pattern
3. Solve individuals pattern
   a. Are we solved the pattern already
   b. Think to relate with old any approach



1.Obesrvation
2. Break it
   a. Print 1 to 15

3. Find the relevant solution for each part
   a. Take a variable name num and increase it

8

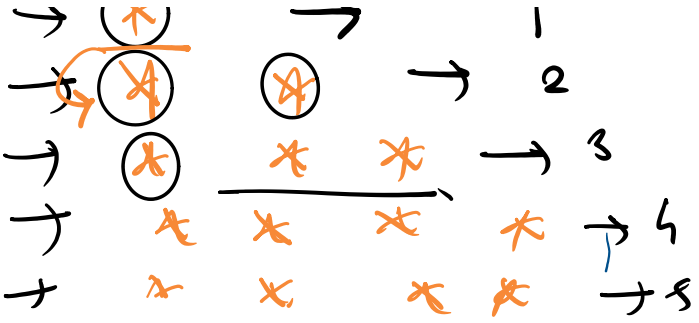
```
for(int i=1; i<=6; i++){
    sout(i);
}
```

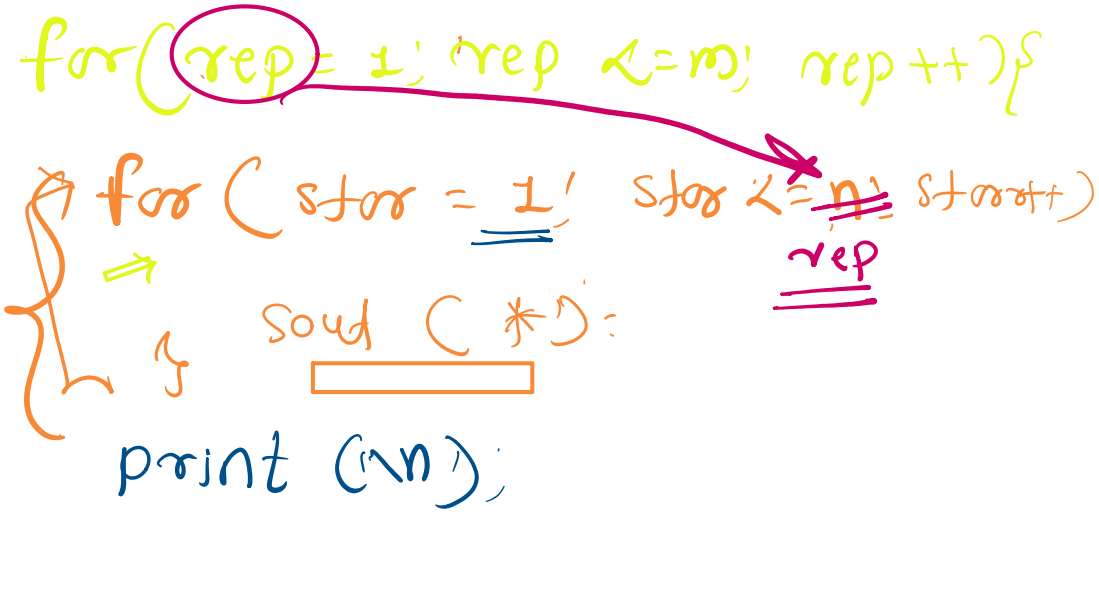
1 2 3 4 5 6.

1
2 3
4 5 6

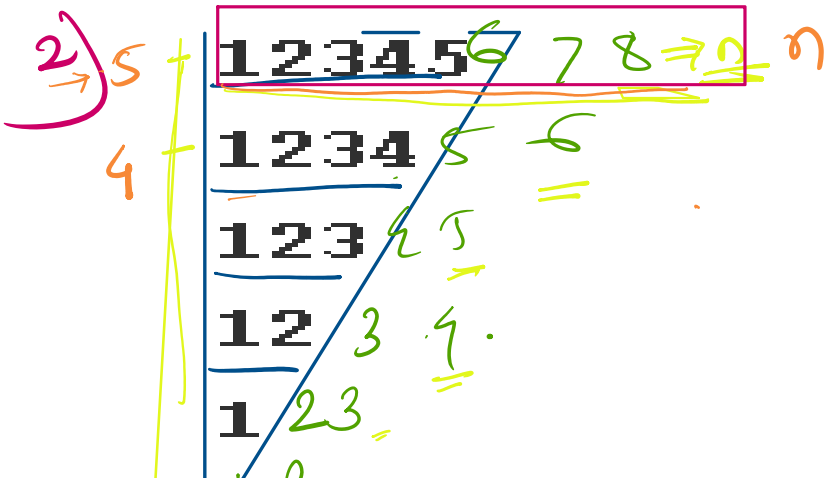
num = 1

row, col → x

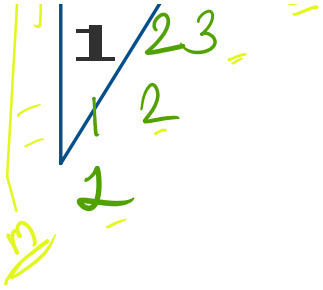





```java
public class PrintNumberRect {
    new *
    public static void main(String[] args) {
        int n = 5;
        int num = 1;
        for(int rep = 1; rep <= n; rep++ ){
            for(int start = 1; start <= rep; start++){
                System.out.print(num + " ");
                num++;
            }
            System.out.println();
        }
    }
}
```

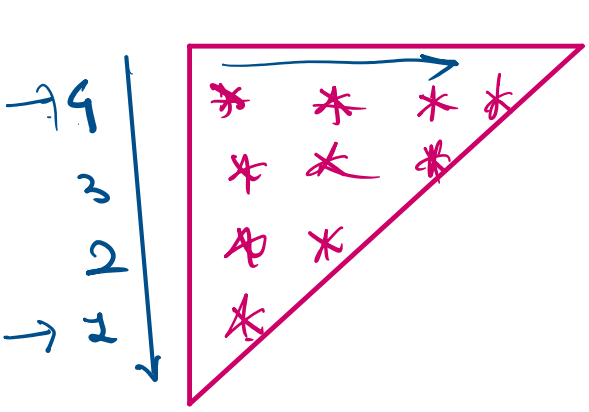


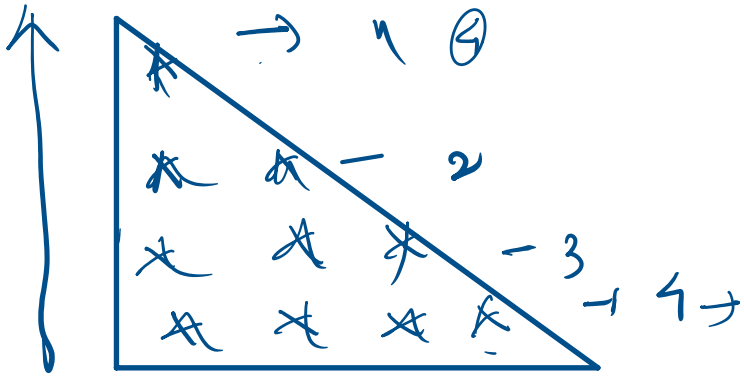
1. Obesrvation
2. Break it
   a. In first it -> we have to print 1 to n
   b. Till m time we have reduce n by 1



1 —— 6.

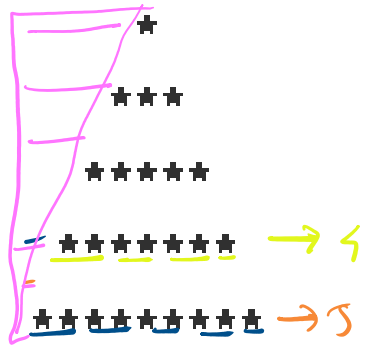
```
for(int num=1; num<=6; num++){
    sout(num);
}
```






```
for(rep = n ; rep>=1; rep--){
    for( num =1; num<=rep; num++
        p (num)
    }
}
```

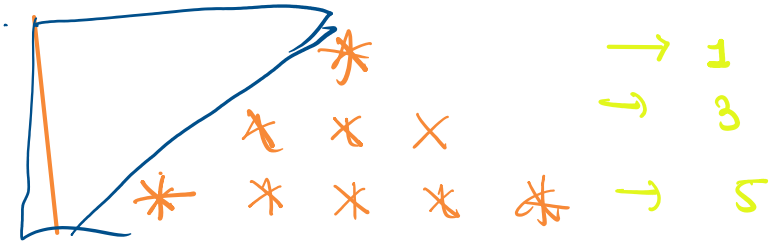
3)



Maximum col / width

1.Obesrvation
2. Break it
   a. We are printing the start 2 * it - 1 / (printing * the in odd manner)
   b. Print the n - it space

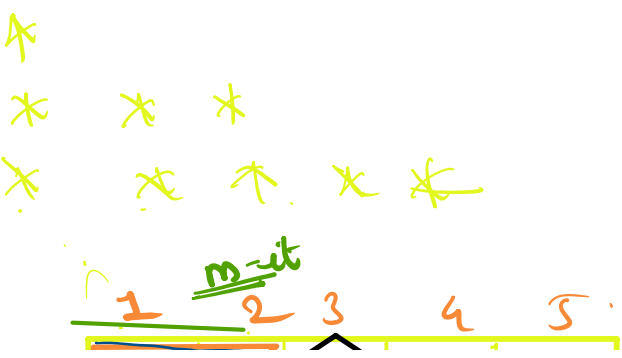
It-> iterations

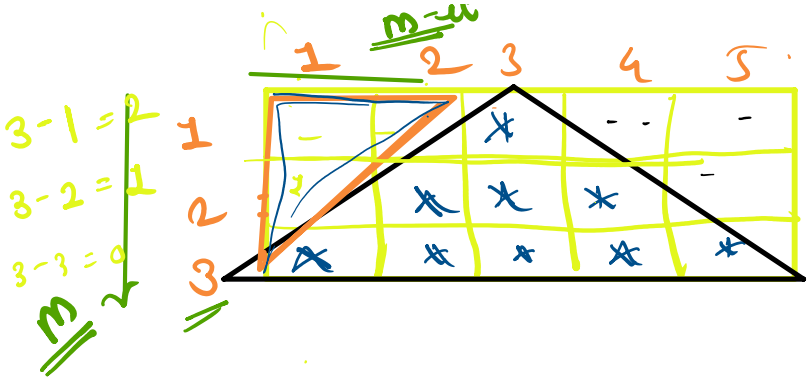


it = 1
it = 2
it = 3 ⇒ 2*it-1
2*3-1
6-1=5

odd


```
{
for(int str=1; str<=2*it-1; str++)
    cout(*);
}
odd+=2
```





⇒ 1 2 = space 3*

⇒ m-i = space

```
for (int space = 1; space <= m; space++)
{
    printf(" ");
}
```

1 to n odd.

```
for (int num = 1; num <= n; num++)
{
}
```

if (num%2) → even
else → odd

```
for (num = 1; num <= n; num+=2)
{
    num
}
```

1 3 5 7 9 . . .

1 3 5 7 9 - - -

num = 1

while (num <= n) {

num += 2

}

```java
public class PrintPyramid {
    new *
    public static void main(String[] args) {
        int n = 5;
        int odd = 1;
        for(int rep = 1; rep <= n; rep++){  // -> odd <= n
            for(int space = 1; space <= n - rep; space++){ // problem b ->
                System.out.print("  ");
            }
            for(int star = 1; star <= odd; star++){  // star  <= 2 * rep - 1;    // Problem
                System.out.print("* ");
            }
            odd += 2;
            System.out.println();
        }
//        for(int i = 1 ; i <= 9; i+=2) System.out.print(i + " ");
    }
}
```

* * * * * * * * *

* * * * * * *

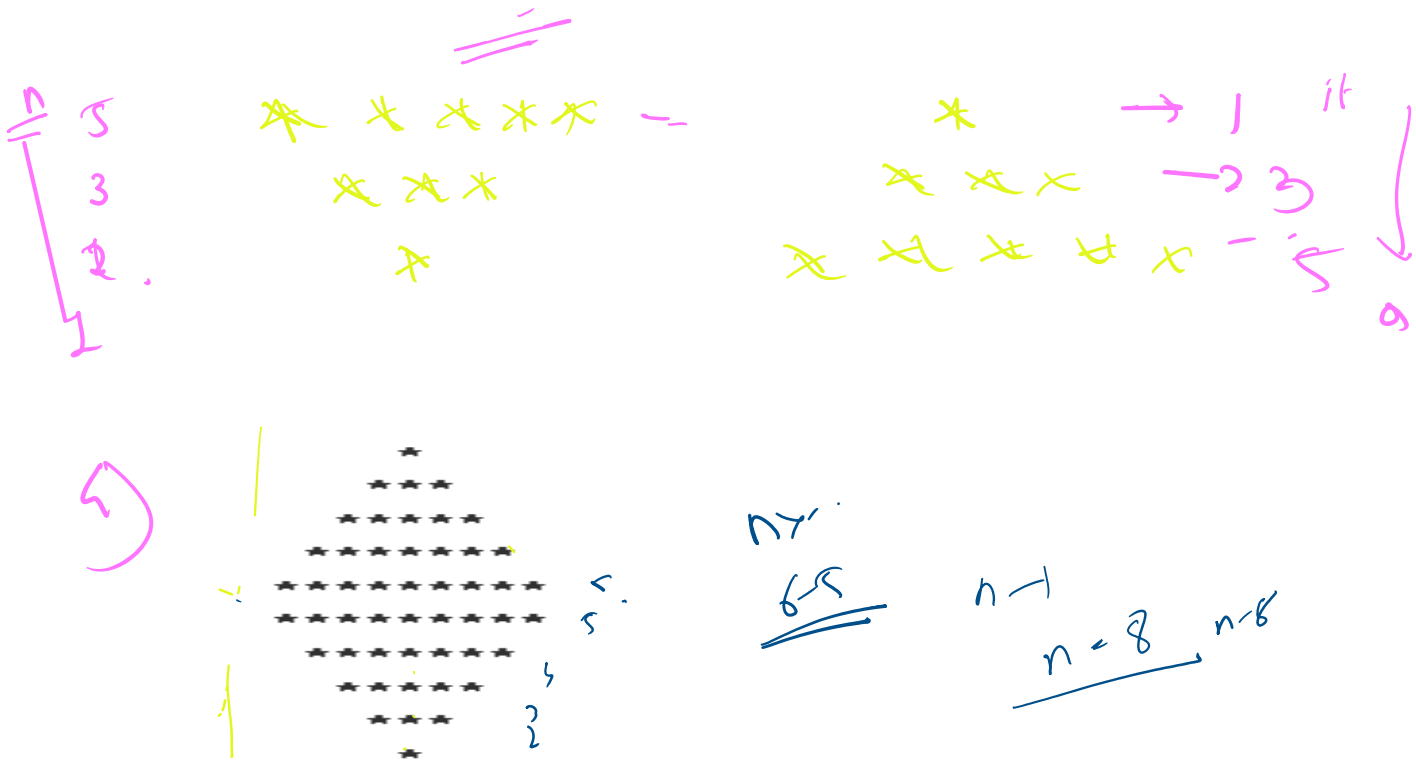
* * * * *

* * *

*

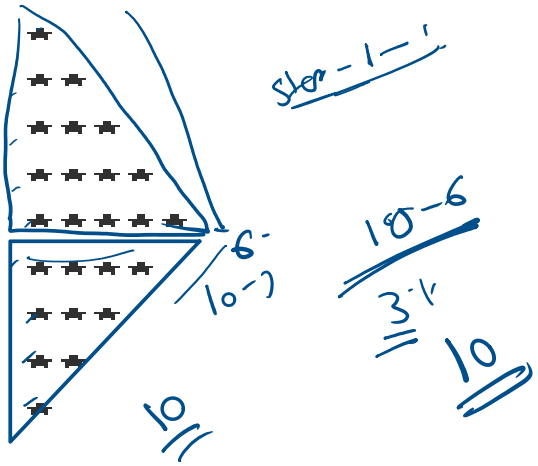
Reverse pyramid

```java
public class ReversetPyramid {
    new *
    public static void main(String[] args) {
        int n = 5;
        for(int rep = n; rep >= 1; rep--){
            for(int space = 1; space <= n - rep; space++){
                System.out.print("  ");
            }
            for(int star = 1; star <= 2 * rep - 1; star++){
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```



```java
public class DiamondPattern {
    public static void main(String[] args) {
        int n = 5;
        int odd = 1;
        for(int rep = 1; rep <= n; rep++){  // -> odd <= n
            for(int space = 1; space <= n - rep; space++){ // problem b ->
                System.out.print("  ");
            }
            for(int star = 1; star <= odd; star++){  // star  <= 2 * rep - 1;    // Problem
                System.out.print("* ");
            }
            odd += 2;
            System.out.println();
        }
        n = 5;
        for(int rep = n; rep >= 1; rep--){
            for(int space = 1; space <= n - rep; space++){
                System.out.print("  ");
            }
            for(int star = 1; star <= 2 * rep - 1; star++){
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

5)




```java
public class halfDimaond {
    public static void main(String[] args) {
        int n = 10;
        for (int rep = 1; rep <= n; rep++) {
            if (rep <= 5) {
                for (int star = 1; star <= rep; star++) {

                    System.out.printf(" * ");    //  -> printing the start

                }
            } else {
                for (int star = 1; star <= n - rep; star++) {   // n - rep + 1;
                    System.out.print(" * ");
                }
            }
            System.out.println();
        }
//        n = 4;
//        for(int rep = n; rep >= 1; rep--){
//            for(int star = 1; star <= rep; star++){   // n - rep + 1;
//                System.out.print(" * ");
//            }
//            System.out.println();
//        }
    }
}
```