⇒ Revision

- Programming language & Operators
- variables
- Conditions and logical operators
- if else
- nested if else
- Switch statement
- Characters and Strings
- * for loops
- while & do while loop
- * Patterns
- * functions (return statement)
- digit traversal (%10 , /10) & number theory
- ** Arrays (Printing , finding & searching , storing info, updating)
- * Brute force (Permutation & Combination)
- * Time & space Complexity

HW_Print the Number Pattern 2

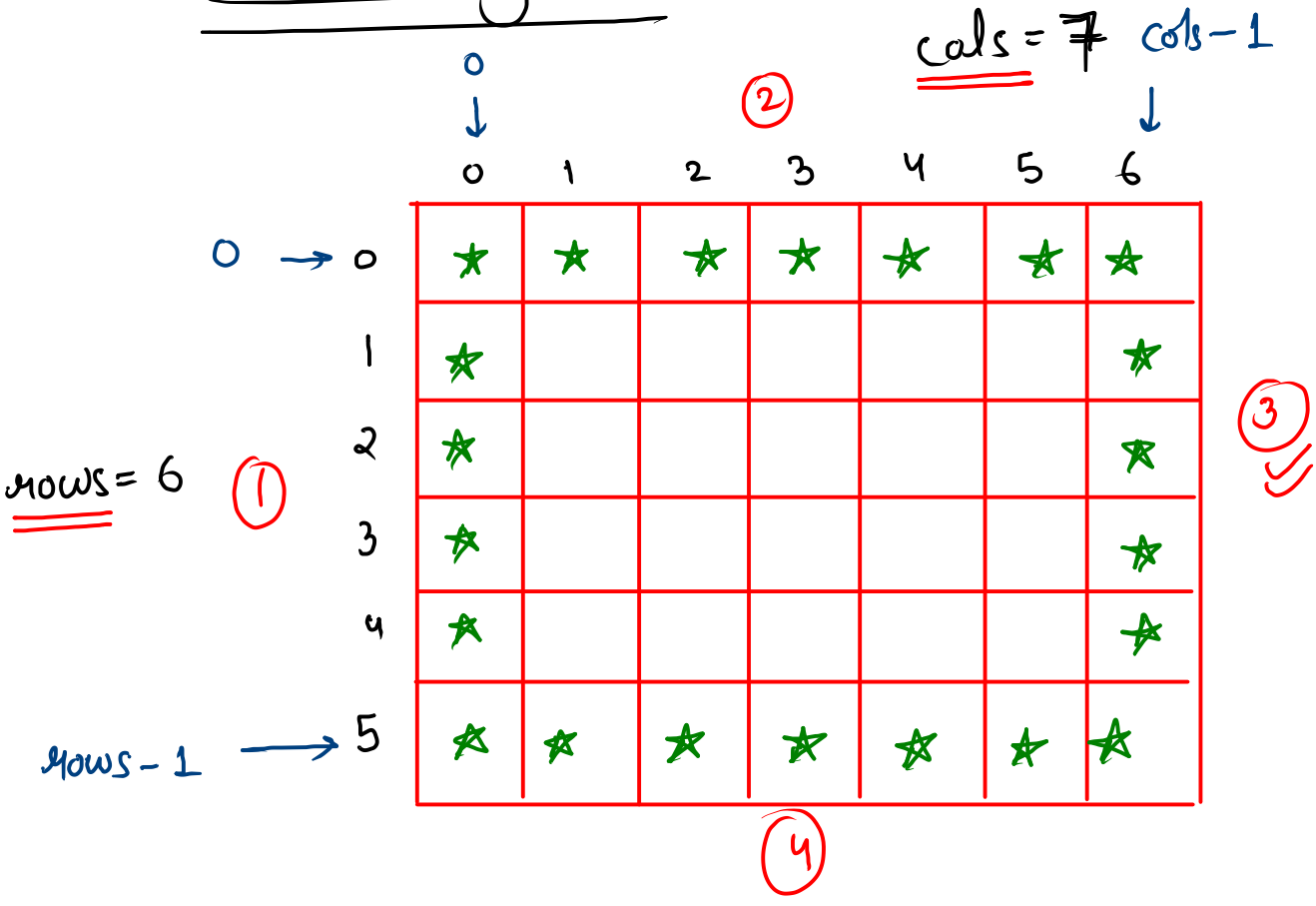
```
      7
    7 14
  7 14 21
7 14 21 28
```

code

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

    int st = 1;
    int sp = n - 1;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < sp; j++) {
            System.out.print("   ");
        }
        for (int j = 0; j < st; j++) {
            System.out.print( k * (j + 1) + " ");
        }
        sp--;
        st++;
        System.out.println();
    }
}
```

⇒ Boundary cases



i == 0
or
i == r-1
or
j == 0
or
j == c-1

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int basic = scn.nextInt();
    char grade = scn.next().charAt(0);

    solve(basic, grade);
}
public static void solve(int basic, char grade) {
    double hra = (basic * 20) * 1.0 / 100;
    double da = (basic * 50) * 1.0 / 100;

    double allow = 0;
    if ( grade == 'A' ) {
        allow = 1700;
    } else if ( grade == 'B' ) {
        allow = 1500;
    } else {
        allow = 1300;
    }

    double pf = (basic * 11) * 1.0 / 100;

    double totalSalary = (basic + hra + da + allow) - pf;
    System.out.println((int)Math.round(totalSalary));
}
```

⇒ Factorial (Imp)

$n! = n * (n-1) * (n-2) * \cdots * 2 * 1$

Logic

```
int ans = 1;
for( int i = n ; i >= 1 ; i--) {

    ans = ans * i ;

}
```

$$^{n}C_{r} \Rightarrow \frac{n!}{(n-r)! * r!}$$

```
a = fact(n);
b = fact(n-r);
c = fact(r);
```

$$ans = \frac{a}{b * c}$$

i/p

$x , y \longrightarrow xy$

$2 , 5 \longrightarrow 25$

ans = $x * 10 + y$

⇒ Reverse a 'n' digit number

num = 1 2 3 4 5 6

ans = 0

dry , num > 0 , int rem = num % 10 , ans = ans * 10 + rem , num = num / 10

steps

1) 123456 > 0 , rem = 6 , ans = 6 ,
2) 12345 > 0 , rem = 5 , ans = 65
3) 1234 > 0 , rem = 4 , ans = 654
4) 123 > 0 , rem = 3 , ans = 6543
5) 12 > 0 , rem = 2 , ans = 65432
6) 1 > 0 , rem = 1 , ans = 654321
7) 0 > 0 ✗

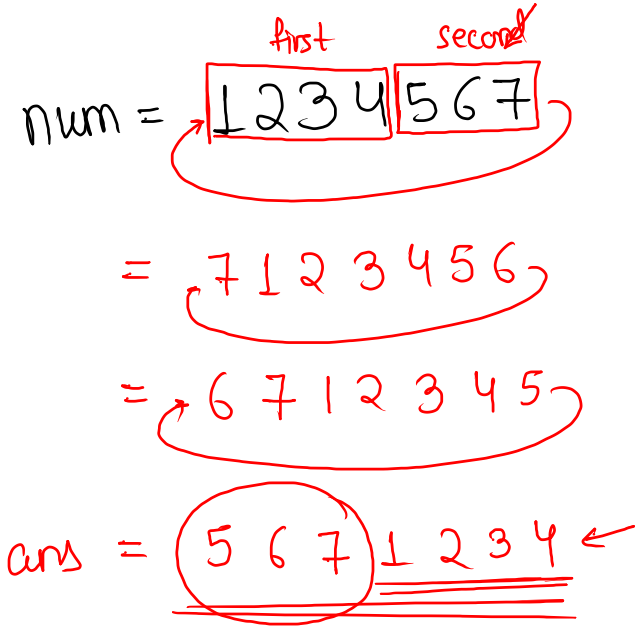
ans = 0

102 > 0, rem = 2, ans = ans * 10 + rem
= 2

10 > 0, rem = 0, ans = 2 * 10 + 0
= 20

1 > 0, rem = 1, ans = 201

Rotate 7-digit number to right by three



int second = num % 1000
int first = num / 1000

ans = 567 * 10000 + 1234
= 5670000 + 1234
= 5671234

ans = second * 10^4 + first

⇒ Prime (n / 1 or n / n)

n ⟶ 2 to n-1 (not prime)

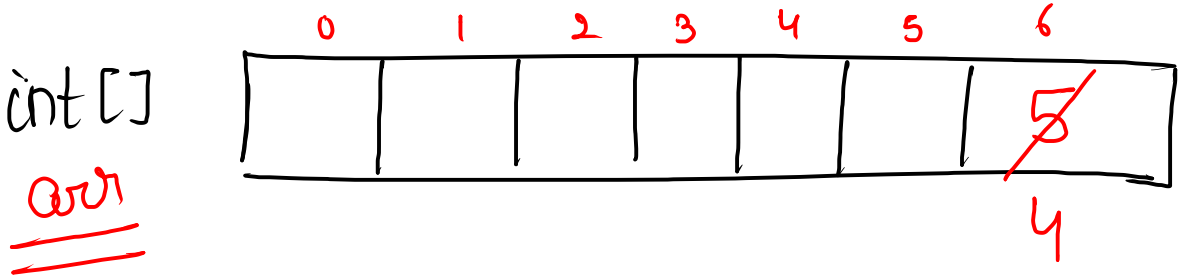
```
for( int i = 2; i < n ; i++) {
        if( n % i == 0) {
            return false;
        }
}
return true;
```

⇒ factors :-

(n) 20 = 2, 4, 5, 10

```
void
for(int i=1; i<=n; i++){
    if(n%i==0){
        syso(i);
    }
}
```

Arrays



// arr.length ; ← size

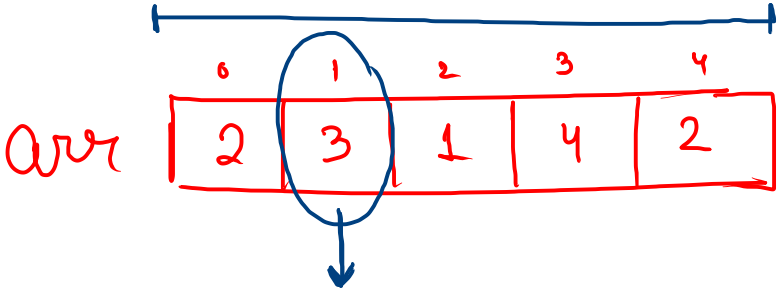
// int n = arr[4] ; ← access

// arr[6] = 4 ; ← update

arr[-1] = 5 ; ← array index out of bound

Product of Elements Except Itself

(Imp)




```
if ( i != j ) {
    product
}
```

for each index, we will traverse from 0 to n-1 (permu).

also skip current element (without rep.)