Digit Traversals

for / while

Given xy Point xy

int x (3) , int y (4) ⇒ int res (34)

res = x * 10 + y

concatenation ⇒ String

resultant = 3 + 4 ≠ 7

= 3 * 10 + 4 * 1 = 34

eg x = 9, y = 0 → xy = 9 * 10 + 0 * 1 = 90

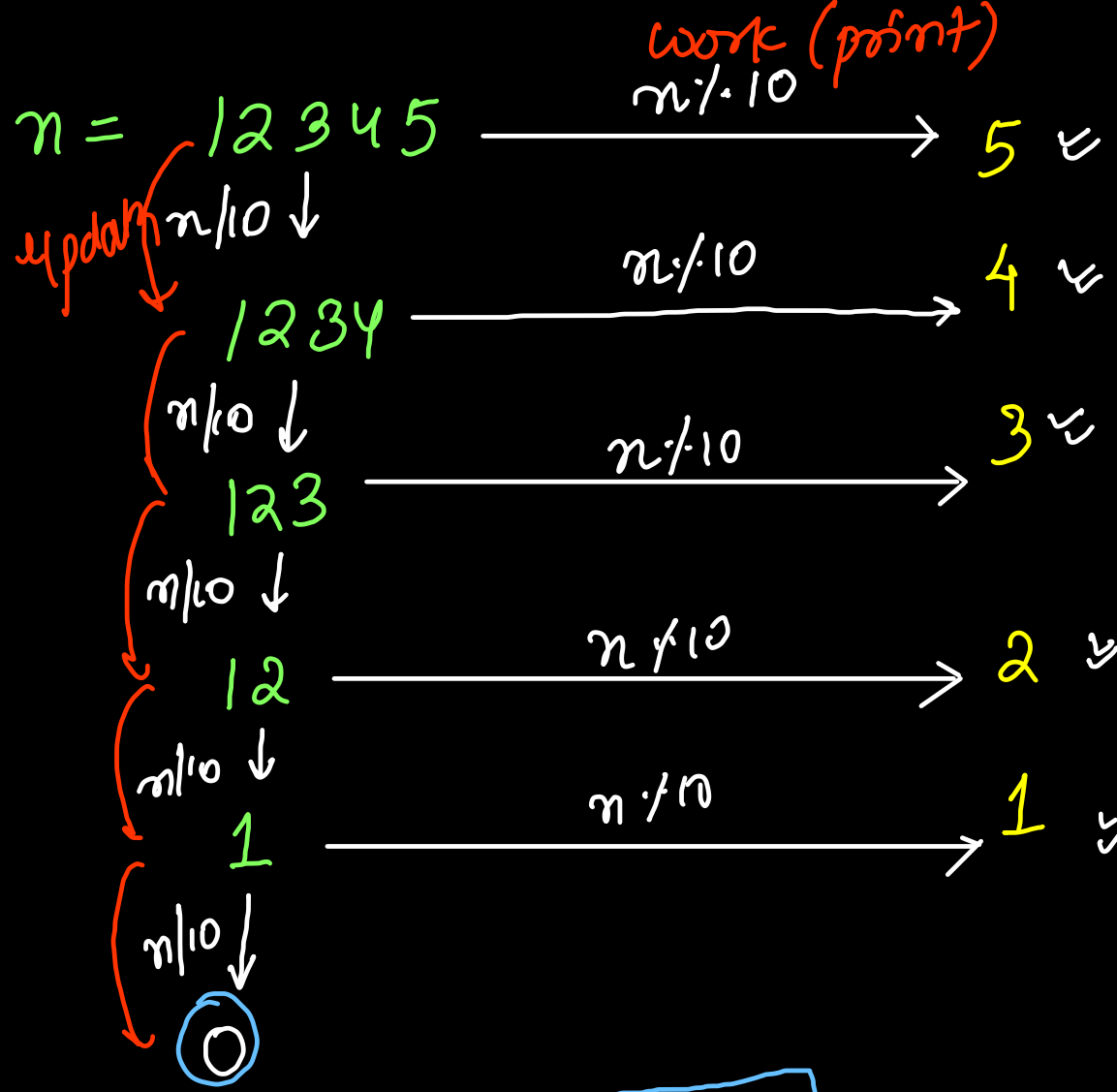
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int tests = scn.nextInt();

    // run the loop for t times
    for(int t = 0; t < tests; t++){
        int x = scn.nextInt();
        int y = scn.nextInt();

        int res = x * 10 + y;
        System.out.println(res);
    }
}
```

tests = 4

	x	y	xy
t=0	4	3	4x10+3 = 43
t=1	1	2	1x10+2 = 12
t=2	3	5	3x10+5 = 35
t=3	5	3	5x10+3 = 53

ten's unit's

Digit Traversal



terminate ⇒ n>0

```
while ( n>0 ){
    int digit = n%10;
    Syso.println(digit);
    n = n/10; or n/=10;
}
```

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

    while(n > 0){
        int digit = n % 10;
        System.out.println(digit);

        n /= 10; // n = n / 10
    }
}
```

While

→ two pointer technique

→ don't know no of iterations

→ conditional based update

Reverse a Number

integer input

integer output

12345 → 54321

not allowed to print digits directly

input = n

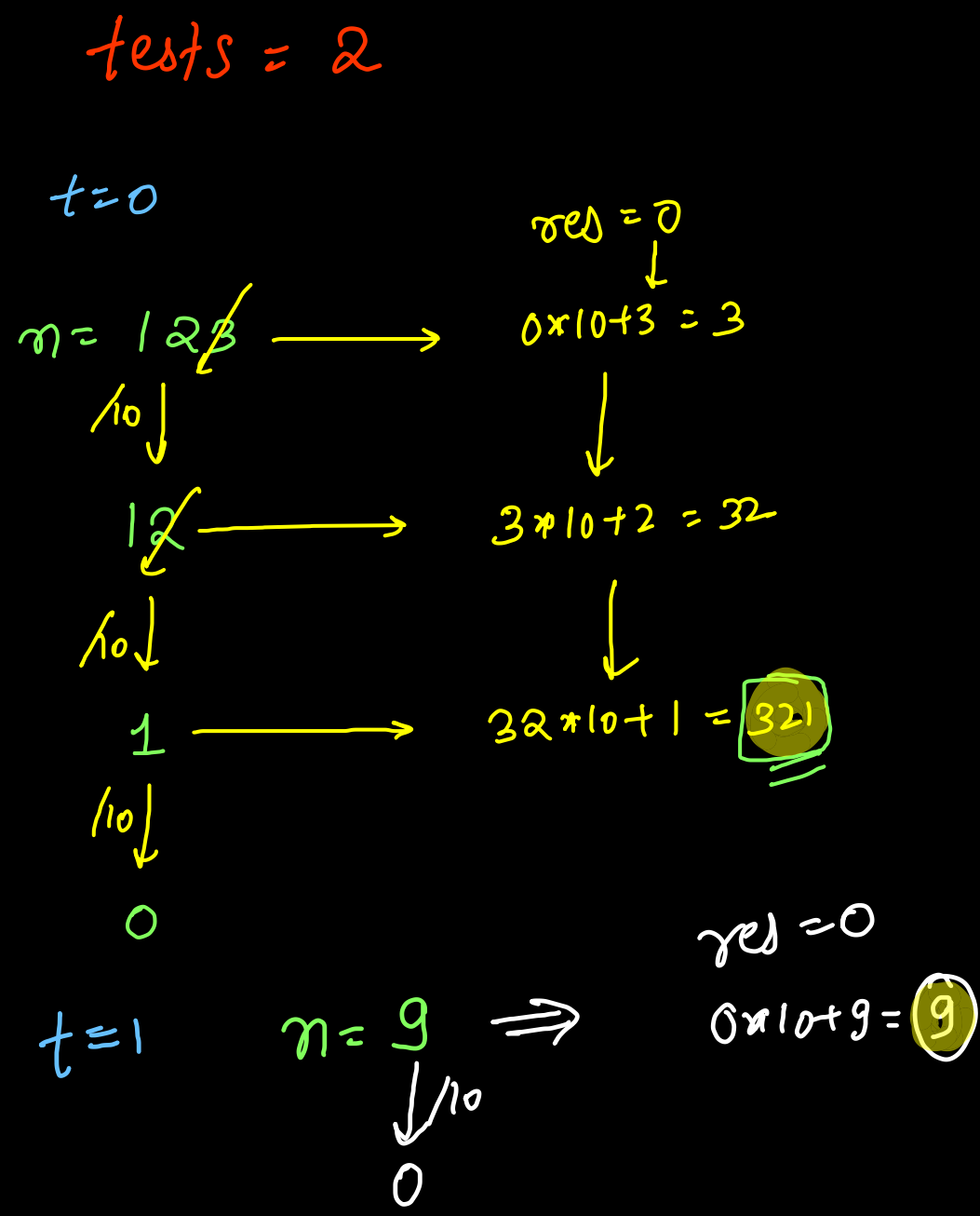
digit

output = 0

n	n%10	digit	output
12345	n%10	5	0×10+5 = 5
1234	n%10	4	5×10+4 = 54
123	n%10	3	54×10+3 = 543
12	n%10	2	543×10+2 = 5432
1	n%10	1	5432×10+1 = 54321
0			

n/10 → 12345 → 1234 → 123 → 12 → 1 → 0

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

    for(int t = 0; t < tests; t++){
        int n = scn.nextInt();
        int res = 0;

        while(n > 0){
            int digit = n % 10;
            res = res * 10 + digit;
            n /= 10;
        }

        System.out.println(res);
    }
}
```



Sample Input 1

```
5
1
2
3
6
7
```

number of digits

Sample Output 1

```
12367
```

integer

From Me no nyzw...

```
int digits = scn.nextInt();
int res = 0;
for(int d=0; d<digits; d++){
    int digit = scn.nextInt();
    res = res * 10 + digit;
}
System.out.println(res);
```

0 * 10 + 1 = 1 ✓

1 * 10 + 2 = 12 ✓

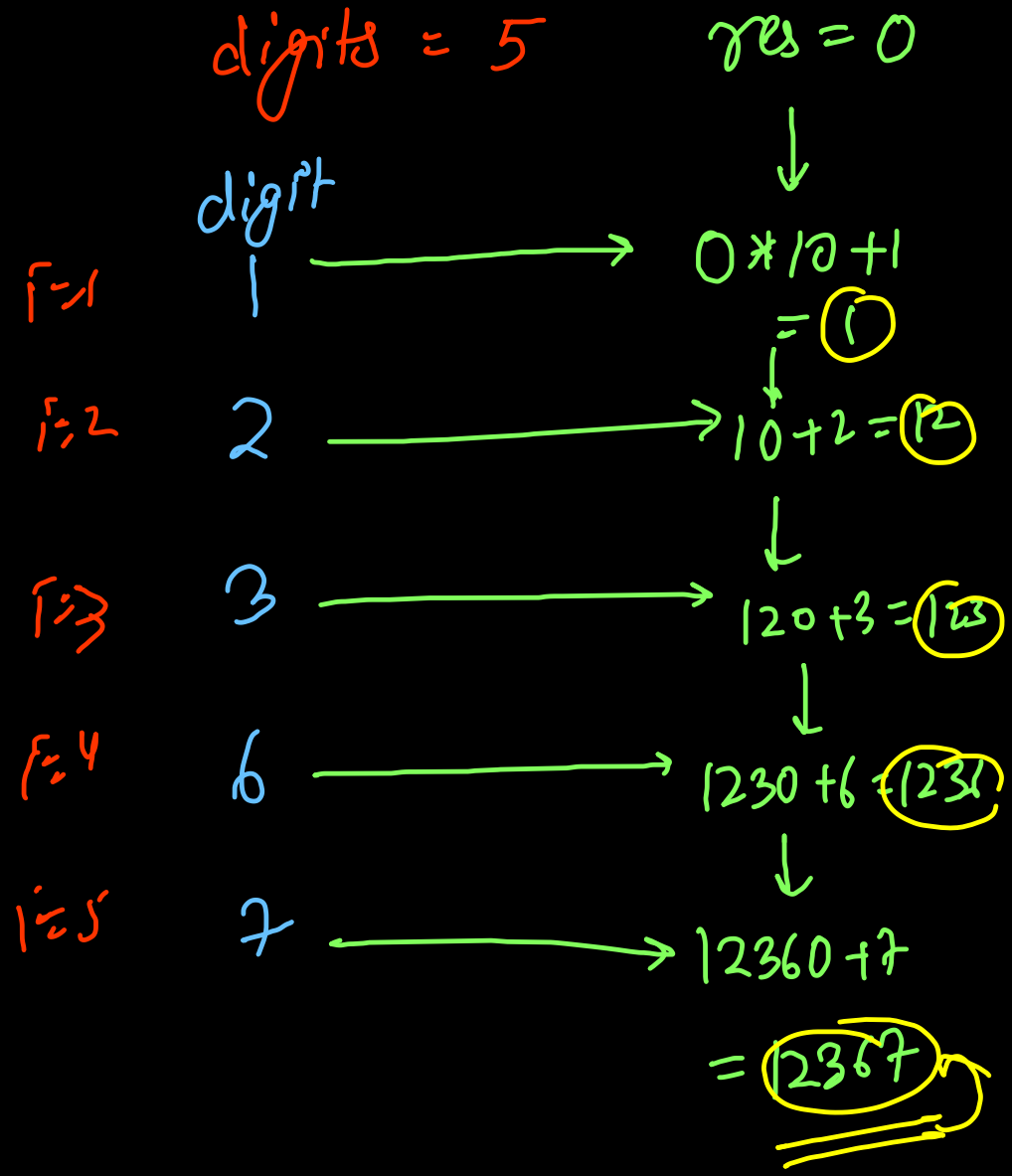
12 * 10 + 3 = 123 ✓

123 * 10 + 6 = 1236 ✓

1236 * 10 + 7 = 12367 ✓

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int digits = scn.nextInt();
    int res = 0;

    for(int i = 1; i <= digits; i++){
        int digit = scn.nextInt();
        res = res * 10 + digit;
    }

    System.out.println(res);
}
```

Interview + Hackerrank (25%)
+ Attendance + weekly contests



Reverse N digit No.

Take a number **n** greater than or equal to **zero** as an integer input.

Then you will be given **n** digits as integer inputs and you have to form a number from it. Print the number formed.

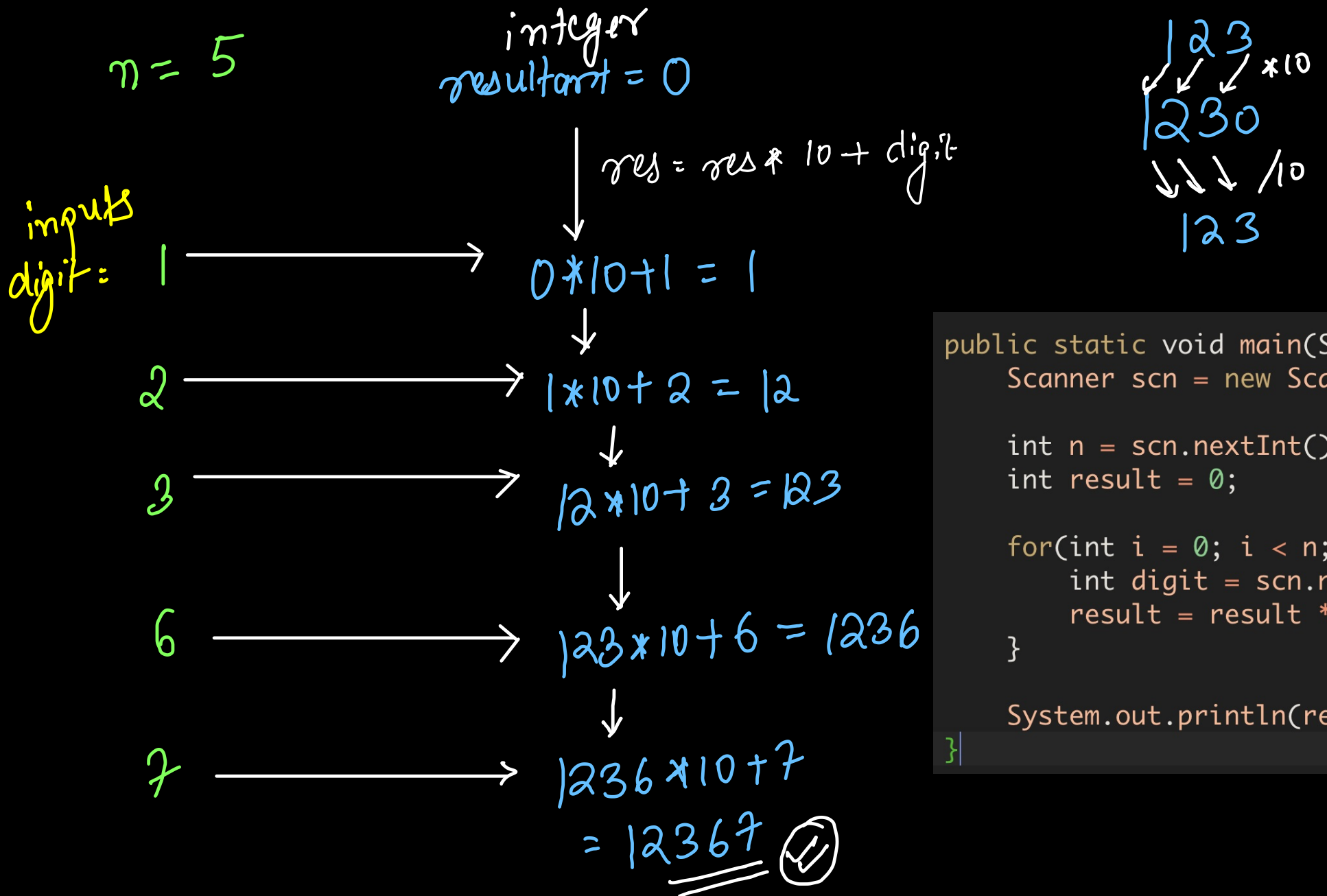
Then you have to **reverse** the digits of this number. And then print the **final reversed number** in the next line.

Sample Input 0

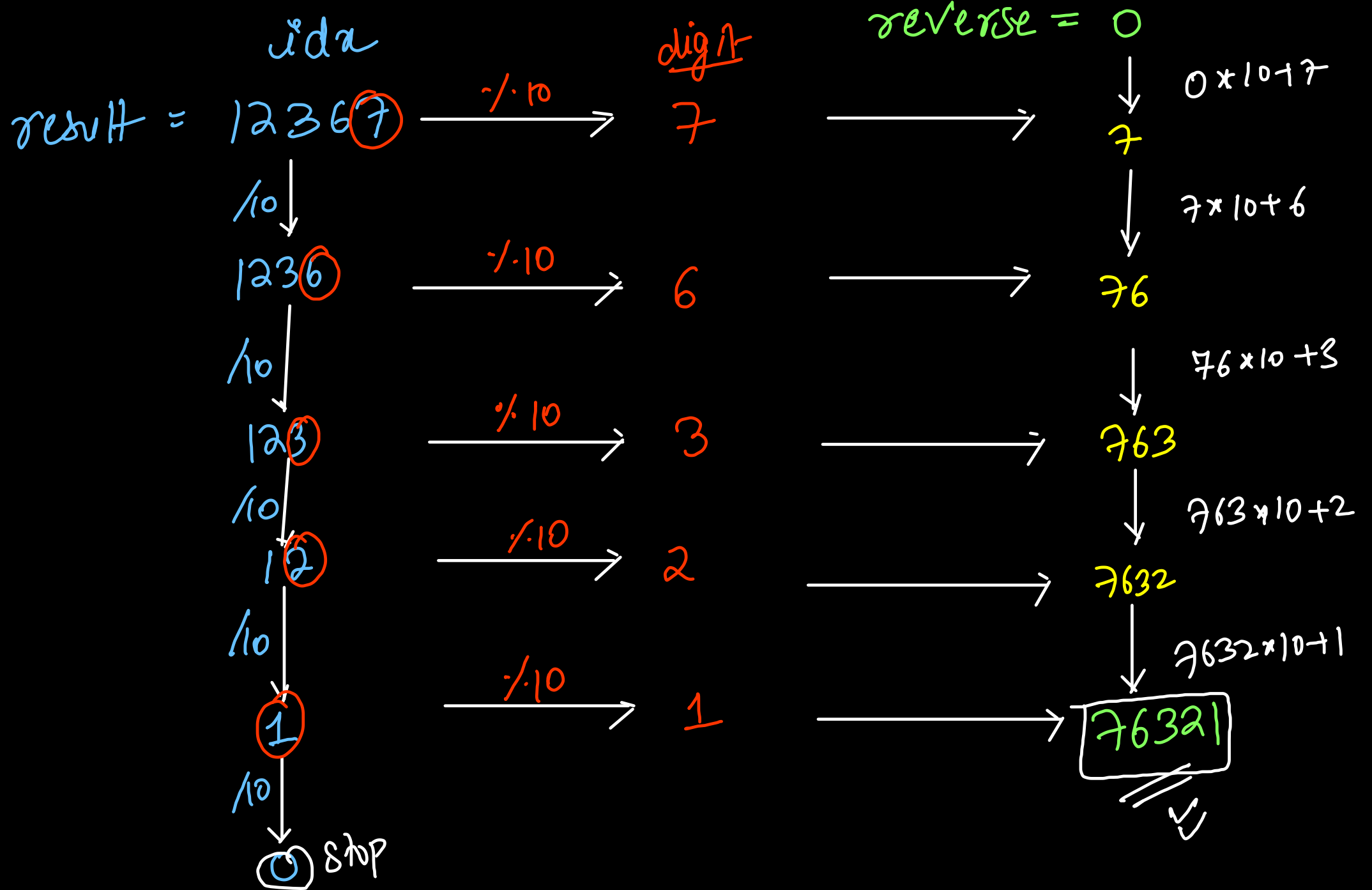
```
3
2
5
6
```

Sample Output 0

```
256
652
```




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

    int n = scn.nextInt();
    int result = 0;

    for(int i = 0; i < n; i++){
        int digit = scn.nextInt();
        result = result * 10 + digit;
    }

    System.out.println(result);
}
```


idx
digit
reverse = 0
result = 12367
%10
7
0*10+7
/10
7
12366
%10
6
7*10+6
/10
76
12363
%10
3
76*10+3
/10
763
12362
%10
2
763*10+2
/10
7632
1
%10
1
7632*10+1
/10
76321
0
stop

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

    int n = scn.nextInt();
    int result = 0;

    for(int i = 0; i < n; i++){
        int digit = scn.nextInt();
        result = result * 10 + digit;
    }

    System.out.println(result);

    int reverse = 0;

    for(int idx = result; idx > 0; idx /= 10){
        int digit = idx % 10;
        reverse = reverse * 10 + digit;
    }

    System.out.println(reverse);
}
```

Forming number from digits

Reversing the number

Armstrong : Sum of cube of digit

✓ 153 = $1^3 + 5^3 + 3^3$

$1 + 125 + 27 = 153$

✓ 370 = $3^3 + 7^3 + 0^3$ $= 27 + 343 + 0$

$= 370$

✗ 214 = $2^3 + 1^3 + 4^3$

$= 8 + 1 + 64 = 73 \neq 214$

```java
public static boolean isArmstrong(int n){
    int sum = 0;

    for(int idx = n; idx > 0; idx /= 10){
        int digit = idx % 10;
        int cube = digit * digit * digit;
        sum += cube;
    }

    if(sum == n) return true;
    else return false;
}

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

    for(int i = 0; i < tests; i++){
        int n = scn.nextInt();
        boolean result = isArmstrong(n);
        System.out.println(result);
    }
}
```

tests = 3

i = 0 : n = 370 ⇒ Sum = 0 + 0^3 + 7^3 + 3^3 = 370

result: true

1 : n = 111 ⇒ Sum = 0 + 1^3 + 1^3 + 1^3 = 3 ≠ 111

false

2 : n = 123 ⇒ Sum = 3^3 + 2^3 + 1^3 = 36 ≠ 123

false

Rotate a Number

1234567

↓ rotate 1st digit

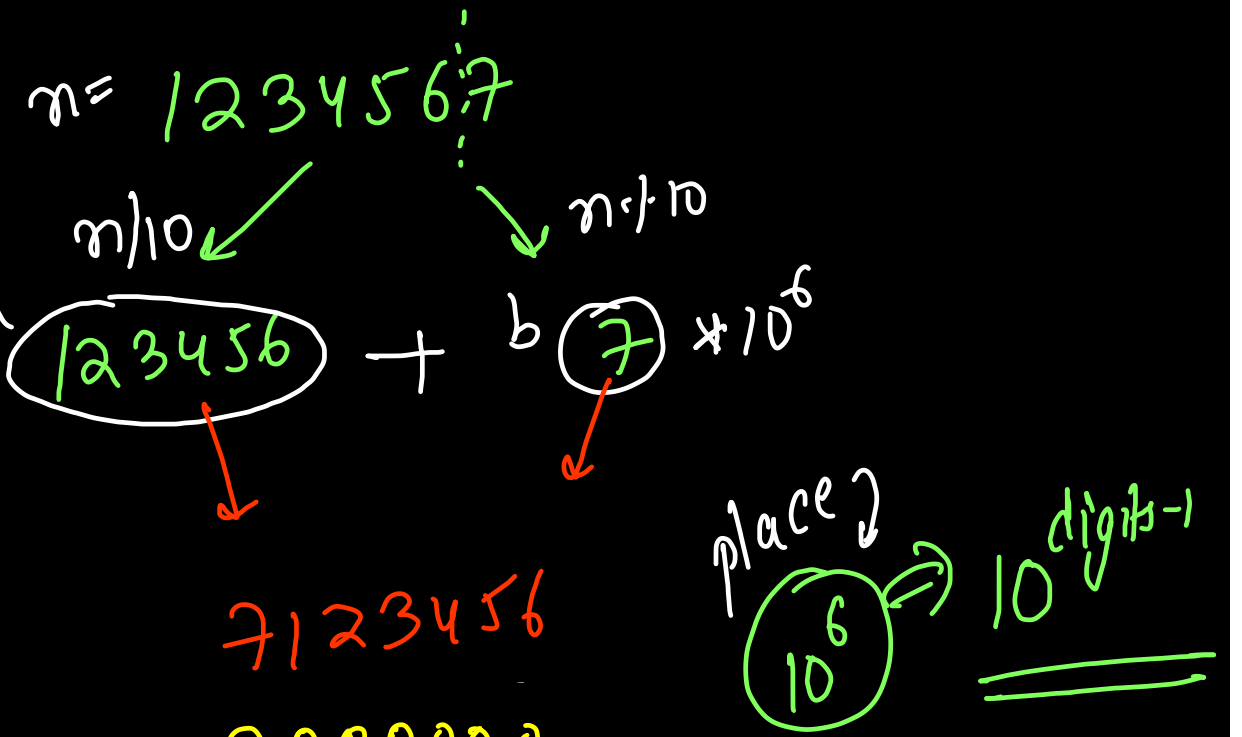
7123456

↓ rotate 2nd digit

6712345

↓ rotate 3rd digit

5671234



1234567

↓ /10

123456

↓ /10

12345

↓ /10

1234

↓ /10

123 → /10 → 12 → /10 → 1 → /10 → 0

count = 0 1

~~1~~ 10

~~2~~ 100

~~3~~ 1000

~~4~~ 10000

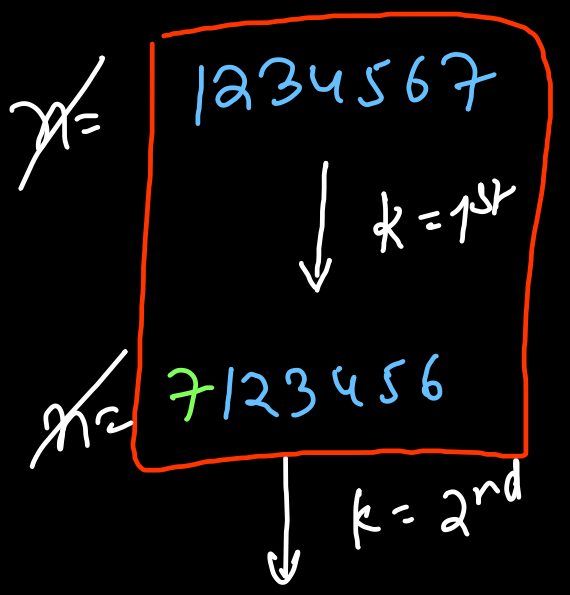
~~5~~ 100000

~~6~~ 7 1000000

```
int place = 1;
for(int idx=n; idx>0; idx/=10){
    place *= 10;
}
```

```java
public static int rotateNumber(int n){
    int place = 1;

    for(int idx = n; idx > 0; idx /= 10){
        place = place * 10;
    }

    place = place / 10; // 10 ^ digits - 1

    int a = n / 10;
    int b = n % 10;

    return (a + b * place);
}

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

    for(int t = 0; t < T; t++){
        int n = scn.nextInt();
        n = rotateNumber(n); //rotate by 1 digit
        n = rotateNumber(n);
        n = rotateNumber(n);
        System.out.println(n);
    }
}
```

123456 + 7×10^6



place = 10^(1234567/10) = 10^6

1234567

123456 | 7

a = n/10 b = n%10

7000000
+123456
7123456

7×6 + 123456

7×10^6 + 123456

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

    for(int t = 0; t < T; t++){
        int n = scn.nextInt();
        int place = 1;

        for(int idx = n; idx > 0; idx /= 10){
            place = place * 10;
        }

        place = place / 10; // 10 ^ digits - 1

        for(int i = 0; i < 3; i++){
            int a = n / 10;
            int b = n % 10;

            n = (a + b * place);
        }

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

place value = 10^6

$10^{0 \times 7 \times 3 \times 4 \times 5 \times 6}$

1234567

a b

rotate by 3

$7 \times 10^6 + 123451$

$= 7123456$

a b

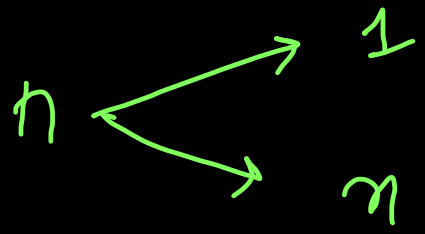
$6 \times 10^6 + 712345$

6712345

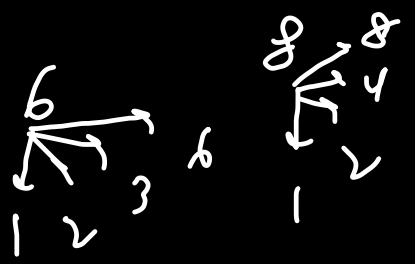
a b

$= 5671234$

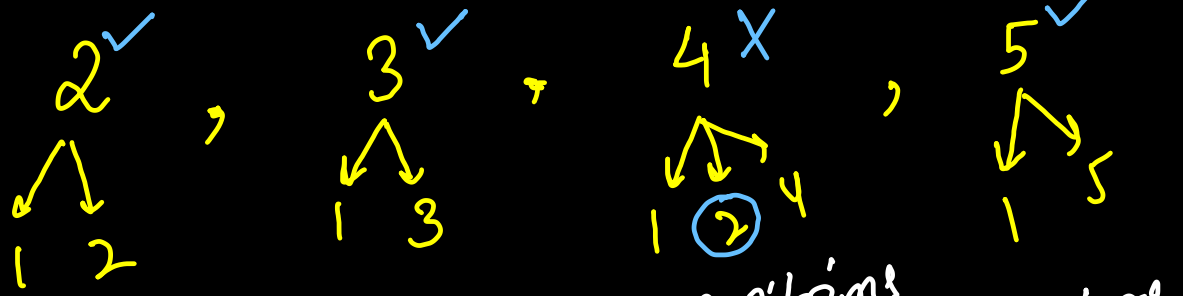
Check Prime No



2 factors / divisors



2 ✓ , 3 ✓ , 4 ✗ , 5 ✓ , 7, 11, 13, 17, 19, 23, 29

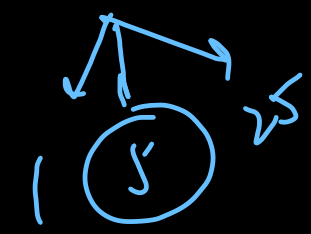


even primes ⇒ only 1 ⇒ ②

~~⇒~~ odd nos will be prime

~~⇒~~ prime (except 2) will be odd

$n = 25$



factor = 2 X

factor = 3 X

factor = 4 X

factor = 5 ✓

No

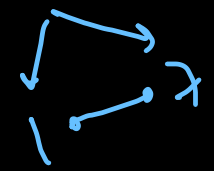
$25 \% 2 \neq 0$

$25 \% 3 \neq 0$

$25 \% 4 \neq 0$

$25 \% 5 = 0$ ✓

$n = 7$



Yes

factor = 2

3

4

5

6

$7 \% 2 \neq 0$

$7 \% 3 \neq 0$

$7 \% 4 \neq 0$

$7 \% 5 \neq 0$

$7 \% 6 \neq 0$

```java
// logic: isPrime
public static String isPrime(int n){
    for(int factor = 2; factor < n; factor++){
        if(n % factor == 0){
            return "No";
        }
    }

    return "Yes";
}

// input, output: main
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);

    int T = scn.nextInt();
    for(int i = 0; i < T; i++){
        int n = scn.nextInt();
        System.out.println(isPrime(n));
    }
}
```

t = 3

i = 0 n = 7

f = 2 7%1 ≠ 0
3 7%3 ≠ 0
4 7%4 ≠ 0
5 7%5 ≠ 0
6 7%6 ≠ 0

Yes

1 n = 25

f = 2 25%2 ≠ 0
f = 3 25%3 ≠ 0
f = 4 25%4 ≠ 0
f = 5 25%5 = 0

No

2 n = 1 f = 2 < 1 X

Yes

```java
// logic: isPrime
public static String isPrime(int n){
    if(n <= 1){return "No";}// corner case

    for(int factor = 2; factor < n; factor++){
        if(n % factor == 0){
            return "No";
        }
    }

    return "Yes";
}

// input, output: main
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);

    int T = scn.nextInt();
    for(int i = 0; i < T; i++){
        int n = scn.nextInt();
        System.out.println(isPrime(n));
    }
}
```

Corner case

$n = 1$ $1 \leq 1$ true

⇒ No

$n = 2$ $2 \leq 1$ false

factor = 2 $2 < 2$: false

loop will not run

"Yes"

Factorization

n = 12 : 1, 2, 3, 4, 6, 12

n = 36 : 1, 2, 3, 4, 6, 9, 12, 18, 36

```
for(int factor = 1; factor <= n; factor++){
        if (n % factor == 0){
            Syso(factor);
        }
}
```

n = 12

12 % ① == 0 ✓

12 % ② == 0 ✓

12 % ③ == 0 ✓

12 % ④ == 0 ✓

12 % 5 ≠ 0 X

12 % ⑥ == 0 ✓

12 % 7, 8, 9, 10, 11 ≠ 0

12 % ⑫ == 0 ✓

```java
public static void printFactors(int n){
    for(int factor = 1; factor <= n; factor++){
        if(n % factor == 0){
            System.out.println(factor);
        }
    }
}

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

module ①

Funcn with return type

⇒ 1 value

Funcn w/o return type

⇒ more than 1 value

Print Unique Prime Factors

$n = 16$

② $16 \% 2 == 0$ ✓ 2 is prime ✓

$16 \% 3 == 0$ X

$16 \% 4 == 0$ ✓ 4 is not prime X

```
for(int f = 2; f <= n; f++){
    if( n%f==0 && isPrime(f)==true)
        Syso(f);
}
```

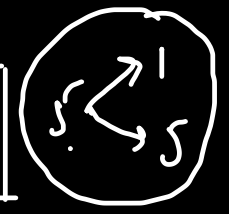
$n = 35$

$35 \% 2 == 0$ X

$35 \% 3 == 0$ X

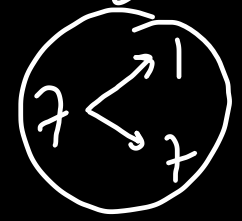
$35 \% 4 == 0$ X

⑤ $35 \% 5 == 0$ ✓



$35 \% 6 == 0$ X

⑦ $35 \% 7 == 0$ ✓



$35 \% 8 \neq 0$

$35 \% 9 \neq 0$

$35 \% 10 \neq 0$

$35 \% 11 \neq 0$

$35 \% 35 == 0$ ✓

prime

35 X

```java
public static boolean isPrime(int i){
    if(i <= 1) return false;

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

    return true;
}

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

    for(int i = 1; i <= n; i++){
        if(n % i == 0 && isPrime(i) == true){
            System.out.println(i);
        }
    }
}
```