* Sum of digits :

eg: 1325

op: 1+3+2+5 = (11)

eg: 565423

op: 5+6+5+4+2+3 = (25)

9254 sum = 0

① 9254 %. 10 = 4
Sum = sum + 4 = 0+4 = 4
num = 9254/10 = 925

② 925 %. 10 = 5
sum = sum + 5 = 4+5 = 9
num = 925/10 = 92

③ 92 %. 10 = 2
Sum = sum + 2 = 9+2 = 11
num = 92/10 = 9

④ 9 %. 10 = 9
Sum = sum + 9 = 11+9 = (20)
num = 9/10 = 0

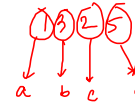
1325 %. 10 = (5)
1325/10 = 132

132 %. 10 = (2)
132/10 = 13

13 %. 10 = (3)
13/10 = 1

1 %. 10 = (1)
1/10 = 0

0 (stop) (until your num = 0)



how to extract each digit

⇒ a + b + c + d

let sum = 0;

1. extract last digit,
const d = num %. 10;
Sum = sum + d;

2. remove last digit,
num = num/10;

3. repeat step 1 and 2 until num becomes 0.

(Digit extraction)

* Reverse of a Number :

eg : 1325

op: 5231

1000 100 10 1

1 3 2 5

⇒ 5*1 + 2*10 + 3*100 + 1*1000

⇒ 5 + 20 + 300 + 1000

= 1325

1325 %10 = 5 → 0*10+5 = 5

132 %10 = 2 → 5*10+2 = 52

13 %10 = 3 → 52*10+3 = 523

1 %10 = 1 → 523*10+1 = 5231

1000 100 10 1

5 2 3 1

⇒ 1*1 + 3*10 + 2*100 + 5*1000

⇒ 1 + 30 + 200 + 5000

⇒ 5231

③ 13 %10 = 3

rev = 52*10 + 3 = 523

13/10 = 1

④ 1 %10 = 1

rev = 523*10 + 1

= 5231

1/10 = 0

let rev = 0;

1. extract last digit,

const d = num %10;

rev = rev *10 + digit;

2. remove last digit,

num = num/10;

3. repeat step 1 and 2 until num becomes 0.

(Digit extraction)

① 1325 %10 = 5

rev = 0*10 + 5 = 0+5 = 5

1325/10 = 132

② 132 %10 = 2

rev = 5*10 + 2 = 52

132/10 = 13

* HCF of two numbers :

Eg: 75, 90

highest common factor

(or) greatest common divisor (HCF / GCD)

→ X is a number such that 75 % X = 0 and 90 % X = 0

and X is the largest possible number

```
for(let num = min(a,b); num > 0; num--) {
    if (a % num == 0 && b % num == 0) {
        console.log(num);
        break;
    }
}
```

* 1st common divisor from backside

75 % 5 = 0

90 % 5 = 0

So 5 is a common divisor/factor

* a, b, GCD <= min(a,b)

→ the common divisor cannot be greater than min(a,b) for sure.

75, 90 — you cannot divide

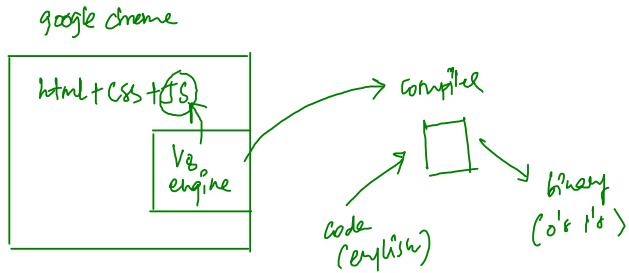
> 75, 75 % 80 = 75

75 % 76 = 75

75 % X != 0 (always)

if X > 75

★ process.stdout.write :



C / C++ / Java / python

mingw | JDK | python

① Install a compiler

JS

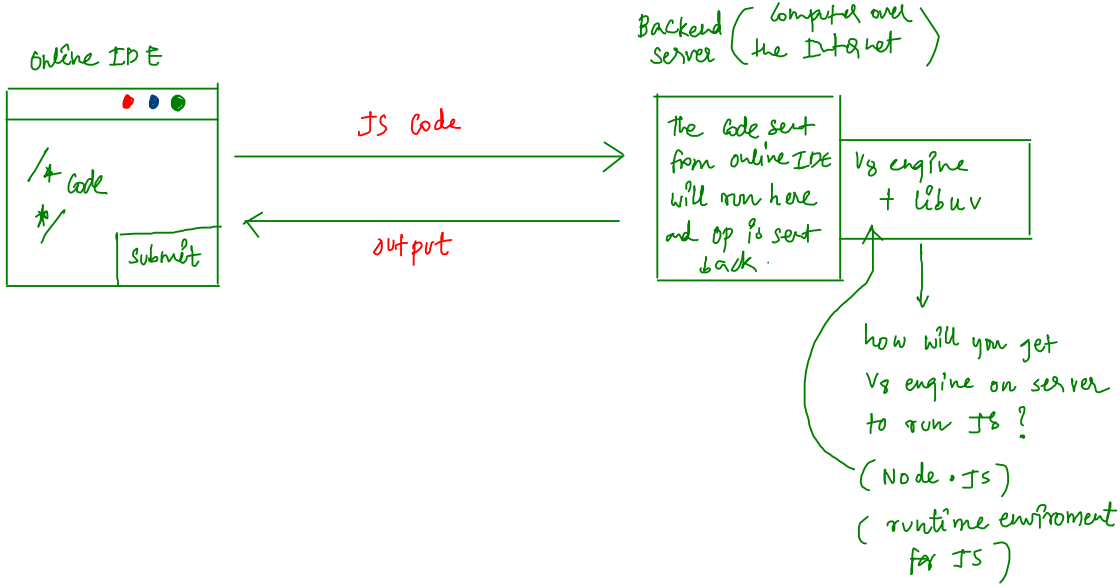
① web browser (chrome)

★ process.stdout.write is not a part of V8 engine hence chrome doesn't understand it. But how are you able to submit it in your online IDE ?

→ because backend server has node.js which provides libuv.

★ process.stdout.write can only print text/strings.

psw(123); ✗ ← number


Online IDE
/* Code */
submit
JS Code
output
Backend server (computer over the Internet)
The code sent from online IDE will run here and op is sent back.
V8 engine + libuv
how will you get V8 engine on server to run JS ?
(Node.JS)
(runtime enviroment for JS)

Improving check prime : $[2, n-1] \rightarrow [2, \sqrt{n}]$

earlier, $n = 13$ → 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

$[2, 12] \simeq n$ checks $\simeq n$ iterations

Now, $n = 13$ → $[2, \sqrt{13}]$ is enough

2, 3 $\simeq \sqrt{n}$ checks $\simeq \sqrt{n}$ iterations

$n = 10^8 \Rightarrow [2, 10^8 - 1] \simeq 10^8$ checks $\simeq 10^8$ iterations → 1 cr

$\Rightarrow [2, \sqrt{10^8}]$ $[2, 10^4]$ $\simeq 10^4$ checks $\simeq 10^4$ iterations → 10 K

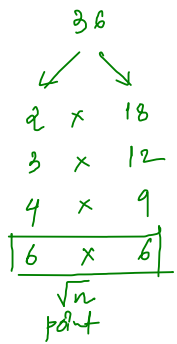
huge improvement

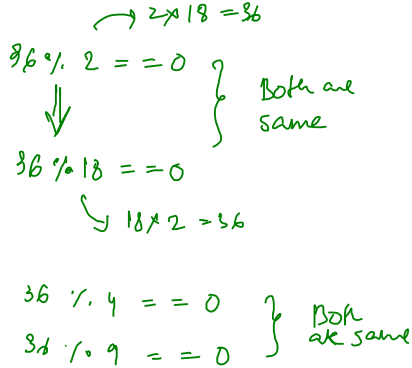
* Why $\sqrt{n}$ is enough ?

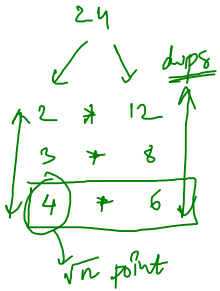
Eg: 36

2, 3, 4, 5, 6, ~~7~~, 8, ~~9~~, 10, 11, ~~12~~,
13, 14, 15, 16, 17, ~~18~~, 19, 20,
21, 22, 23, 24, 25, 26, 27,
28, 29, 30, 31, 32, 33, 34,
35

duplicates







24 % 2 } same
24 % 12

24 % 3 } same
24 % 8

$\sqrt{24} = 4.\ldots$
$\simeq 4$