

(Wednesday)

① → Sum of Digits

let $sum = 0 +$

① $num = 456$

$int\ digit = 456 \% 10 = 6$

by doing this we get
last digit

② $num = 456 / 10 = 45$

$digit = 45 \% 10 = 5$

$num = 45 / 10 = 4$

③ $digit = 4 \% 10 = 4$

$num = 4 / 10 = 0$

↳ stops

→ $num \neq 0$

↳ $digit = n \% 10$

↳ $sum = sum + digit$

↳ $num = num / 10$

② → Return the smallest digit from the Number.

1/p Num = 420

4
2
0

0/p: → 0

Rule:

Integer → Range \textcircled{x} → \textcircled{y}
 min max
 ↓ ↓
 INT-MIN INT-MAX

5 7 9 4 6

if

① $5 < \text{INT-MAX}$

int minimum = INT-MAX

= ~~5~~ $\textcircled{4}$

- ① To find out Smallest number we initialize with INT-MAX
- ② To find out Largest number we initialize with INT-MIN

③ Include header file

#include <limits.h>

int min = INT-MAX; ← compare.

num = 5289

digit = num % 10 = 5289 % 10 = $\textcircled{9}$

num = num / 10 = 5289 / 10 = 528

③

$$\text{digit} = 528 \% 10 = 8 \rightarrow [\text{Compare with min Val.}]$$

$$\text{num} = 528 / 10 = 52$$

$$\text{digit} = 52 \% 10 = 2$$

$$\text{num} = 52 / 10 = 5$$

$$\text{digit} = 5 \% 10 = 5$$

$$\text{num} = 5 / 10 = 0 \rightarrow \text{Stop}$$

→ Inbuilt function

$\text{min}(a, b);$

$\text{max}(a, b);$

} we can use these directly
to get min, max.

→ Note:

- Find out the best alternative other than '%'. because % is an heavy operation, % is not a best practice.

③ → check if a number is even or odd.

Even
2
4
6
8
10
multiples of 2
 $12 \rightarrow \text{rem} \rightarrow 0$
 $\% 2 == 0$

odd

1
3
5
7
9

$12 \rightarrow \text{rem} \rightarrow 1$
 $\% 2 == 1$

if $(\text{num} \% 2 == 0)$
{
"Even"
}
else
{
"Odd"

④

→ Alternate method

2 → 00000010

4 → 00000100

6 → 00000110

8 → 00001000

10 → 00001010

Even: last bit 0 & 1 → 0

odd: last bit 1 & 1 → 1

↓
odd

0	1
<u>1 1</u>	<u>1 1</u>
<u>0</u>	<u>1</u>

Note:

- The Best approach is using Bit-wise approach. we can use this in any problem because there is no best approach other than bit wise.

☆☆☆

→ Reverse a Number: (Asked in Online Test, Interview)

④

I/p

→ n = 123

O/p → 321

I/p → n = 421

O/p → 124

I/p → n = 479

O/p → 974

Approach:

① n = 123

n % 10 = 3

n = 123
10 = 12

5

$$n \% 10 = 12 \% 10 = 2$$

$$n / 10 = 12 / 10 = 1$$

$$n \% 10 = 1 \% 10 = 1$$

$$n / 10 = 1 / 10 = 0 \rightarrow \text{stops}$$

321

$$300 \rightarrow 3 \times 10^2$$

$$+ 20 \rightarrow 2 \times 10^1$$

$$+ 1 \rightarrow 1 \times 10^0$$

int ans = 0;

$$\textcircled{1} 0 \times 10 + 3 = 3$$

$$\textcircled{2} 3 \times 10 + 2 = 32$$

$$\textcircled{3} 32 \times 10 + 1 = 321$$

② int ans = 0;

$$n = 576$$

$$\% 10 \rightarrow 6$$

$$0 \times 10 + 6 \rightarrow 6$$

$$n = 576 / 10 = 57$$

$$\% 10 = 7$$

$$6 \times 10 + 7 = 67$$

$$n = 57 / 10 = 5$$

$$\% 10 = 5$$

~~$$57 \times 10 + 6 = 576$$~~

$$67 \times 10 + 5 = \underline{675}$$

$$n = 5 / 10 = 0$$

\rightarrow stop

⑥

while (num != 0)

{

int digit = num % 10;

ans = ans * 10 + digit;

num = num / 10;

}

return ans;

alternate:

for C ; num != 0; num = num / 10)

{

int digit = num % 10;

ans = ans * 10 + digit;

}

return ans;

→ Reverse an Integer.

⑤

Algorithm:

↳ ① 0 wala case handle krdo

② Sign ki info flag me rakhlo

③ Sign remove krdo [Use: abs(num) for removing sign]

④ num reverse krdo

⑤ Reverse no ke andhe sign lagado

⑥ return krdo

Ex: n = -123

if (num < 0)

flag = 0

flag → variable.

1 → +ve

0 → -ve

flag = 0

n = 123

ans = 321

ans = -321 → return

→ Prog: (Dry Run):

if (n == 0) return 0;

① n = -56

boolean flag = 0;

② flag = 0 ✓

if (n > 0)

~~return~~ flag = 1;

✓

n = 123456789

flag = 0

flag = 1

n = abs(n);

n = abs(-56) = 56

long long int ans = reverseNumber(n);

ans = 65

ans = 9... -1

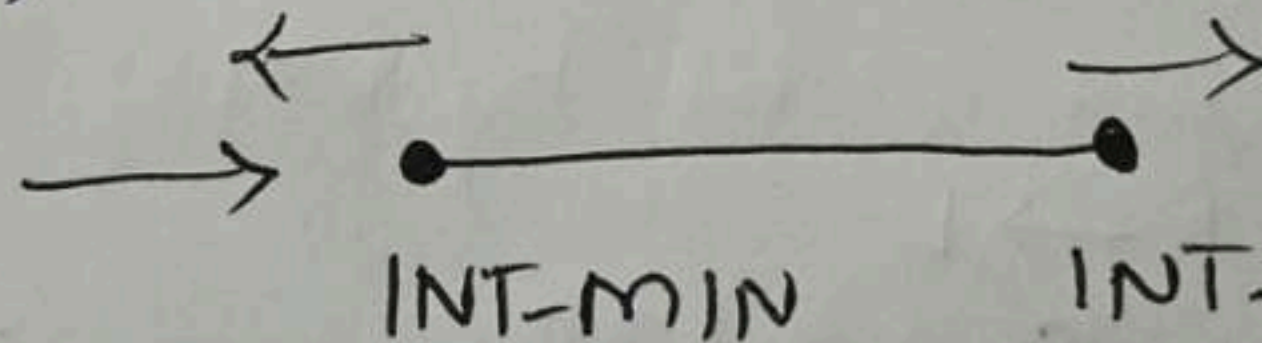
if (flag == 0)

ans = 0 - ans;

✓

ans = -65

// range check



ans > INT-MAX

if (ans > INT-MAX || ans < INT-MIN)

return

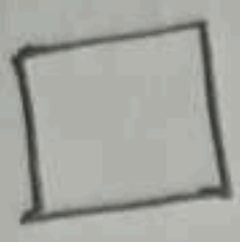
return 0;

return ans;

-65

}

8

 ~~int~~ long long int
Value

abs → Exact value either it is +ve or -ve.

Ex: $\text{abs}(5) = 5$

$\text{abs}(0) = 0$

$\text{abs}(-5) = 5$

Can we make our own
function if we have
any confusion.

int absolute value (int n)

{

if (n > 0)

return n;

else

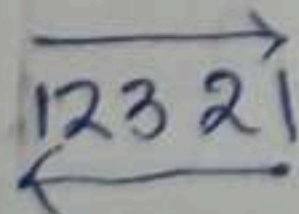
return (0 - (n));

}

→ Palindrome Number: (Asked in Online Test, Interview)

⑥

$L \rightarrow R = R \rightarrow L$
(reverse)

Ex: 

① reverse the Given number

② Compare with that reverse. (org == rev)

↳ then it is a palindrome

⑨

String isPalindrome (int num) {

int original = num;

int rev = reverseNumber(num);

if (original == rev) {

return "Yes"

}

else {

return "No";

}

}

→ Number is a power of 2 or Not

⑦

0 → does not have Set bit

1

$2^0 \rightarrow 1 \rightarrow 0000 \dots 0001$

$2^1 \rightarrow 2 \rightarrow 0000 \dots 0010$

$2^2 \rightarrow 4 \rightarrow 0000 \dots 0100$

$2^3 \rightarrow 8 \rightarrow 0000 \dots 1000$

$2^4 \rightarrow 16 \rightarrow 0000 \dots 10000$

2^x

Having only

1 set bit

$1/p \rightarrow n$

→ No of set bit = 1 → Can be represented as 2^x (or) power of 2

Q

→ How can I find Single Set bit

result as 0
- $n \& (n-1)$ gives ~~[00000000]~~. If 0 then it is power of 2

$n=8 \rightarrow 00000001$

~~$n-1$~~ $(n-1)=8-1=7 \rightarrow 00000111$

00000000

$n=16 = 0000000000000001$

$n=15 = 0000000000000111$

0000000000000000

→ If we cant makeup formula then

Hw

32 bit

→ count Set bit by using $>>$ operator
with $O(1)$ time

① → Check if a Number is prime or Not.

* Prime Number

↳ have factors 1 and itself

13 → 1, 13

2 → 1, 2

* 1 — Neither prime nor composite

Prog:

```
if (num == 1)
```

```
{
```

```
    return "No";
```

```
}
```

```
if (num == 2 || num == 3)
```

```
{
```

```
    return "Yes"
```

```
}
```

```
int start = 2;
```

```
int end = n-1;
```

```
for (int i = start; i <= end; i++)
```

```
{
```

```
    if (num % i == 0)
```

```
    {
```

```
        return "No";
```

```
    }
```

```
}
```

```
    return "Yes";
```