Armstrong number:

Ex: n = 153

= $1^3 + 5^3 + 3^3$

= 1 + 125 + 27

= 153

∴ 153 is an armstrong number.

n = 123

= $1^3 + 2^3 + 3^3$

= 1 + 8 + 27

= 36

∴ 123 is not an Armstrong number.

$(6152)_{10}$ ⇒ 6000 + 100 + 50 + 2

$10^3 \times 6$ $10^2 \times 1$ $10^1 \times 5$ $10^0 \times 2$

1000   100   10   1 (×10, ×10, ×10)

2516 ⇒ 2000 + 500 + 10 + 6

2 → 3 → 2000

5 → 2 → 500

Q. Reverse the no:

int n = 34921

o/p: 12943.

val = 1 2 9 4 3

return

Q. Take i/p from user (1 no.). If it an Armstrong?

num = 125

$1^3 + 2^3 + 5^3$

1 + 8 + 125 ∴ It's not an Armstrong

134

153 ⇒ Armstrong.

Factorial:

int n = 5;

5 * 4 * 3 * 2 * 1 ⇒ 120 ⇒ o/p.

int n = 6;

6 * 5 * 4 * 3 * 2 * 1 ⇒ 720 ⇒ o/p.

n = 8

o/p:

1+2+3+4+5+6+7+8

odd nos: int n = 25;

2)25(12 − 24 = 1

2)26(13 − 26 = 0

2)45(22 − 44 = 1

```
if(n%2==1){
    sout(odd);
}else{
    sout(even);
}
```

i = ~~1~~ ~~2~~ ~~3~~ 5

sum = ~~0~~ ~~1~~ 4

1+3+5+7+9 … 99

Q. sum of odd nos in the range 1 to 100.

```
int sum=0;
for(int i=1; i<=100; i++){
    if(i%2==1){
        sum=sum+i;
    }
}
sout(sum);
```

1   3

1. Print 1 to 100.
2. Print odd nos.
3. Sum of them.

In the medieval age, there were 3 kingdoms A, B, and C. The army of these kingdom had N_A, N_B, and N_C soldiers respectively.

You are given that an army with X soldiers can defeat an army with Y soldiers only if $X > Y$.

An army is said to be **dominant** if it can defeat both the other armies **combined**. For example, kingdom C's army will be dominant only if $N_C > N_A + N_B$.

Determine whether any of the armies is **dominant** or not.

N_A	N_B	N_C
1000	200	150

Chef has a bucket having a capacity of K liters. It is already filled with X liters of water.

Find the **maximum** amount of extra water in liters that Chef can fill in the bucket without overflowing.

K = 25

X = 10

o/p: ?