nxn pattern:

```
int n = 4;
for(i=1; i≤n; i++){
    for(j=1; j≤n; j++){
        sout(*);
    }
    sopen()!
}
```

i=1	n
i=2	n
i=3	n
i=4	n
⋮	⋮
i=n	n

$n \times n = O(n^2)$.

Rules:

i) Drop the non-dominant term.

ii) Drop the constants.

iii)

T.C → Big Oh, Big Omega, Big zeta

TC: $O(num)$ — 10, 10,00000

```
int x = 0;
int y = 1;        } k1
for(i=0; i≤num; i++){
    x += num;     } k2      k2 x num
    y++;
}
int sum = 10;
sout(sum);        } k3
```

Formula to calculate sum of n natural no.

num = 6   o/p: 21

$\frac{n \times (n+1)}{2} = \frac{6 \times 7}{2} = \frac{42}{2} = 21$

for(

num = 5   o/p: 15

Algo 1:

```
int num = 5;
int sum = 0;
for(i=1; i≤num; i++){
    sum = sum+i;
}
sout(sum);
```

Algo 2:

$int\ x = \frac{n \times (n+1)}{2} = \frac{5 \times 6}{2} = \frac{30}{2}$

sout(x);

log?

16

$2^3 = 8$

$\log_2 8 = 3$

$\log_2 16 = 4$

```
sum = 0
for(i=0; i<10; i++){
    sum += i;
}
```

T.C = The amount of
= No of the statem is executed.

Time complexity:

→ Independent of machine

→ Direct relation with increase in input

Question:

2 i/p: a and b. Add a and b, c ... a.b.c simultaneously. d

d - a+b+c.

c = a + b

d = a * b * c

e = d - (a+b+c)

f, d

h

DRY principle:

```
for(i=0; i<n; i++){
    sout(i);
}
for(i=0; i<n; i++){
    sout(i);
}
for(i=0; i<n; i++){
    sout(i);
}
```

```
for(i=0; i<n; i++){
    sout(i);
}
```

NON-modular

functions / methods:

x = 5 → $f(x^2) = 25$

$f(x) = 5$

syntax for methods:

```
public static return-type name (input){

}
```

Examples :

```
int sum = 0;
int prod = 1;
for(i=0; i<n; i++){ → O(n)
  sum = sum + i;
}
for(i=0; i<n; i++){ → O(n)
  prod = prod * i;
}
```

T.C : O(n) + O(n)
= O(n).

```
for(i=1; i<=n; i++){
  for(j=1; j<=i; j++){
    sout(*);
  }
  sopln();
}
for(i=0; i<n; i++){
  sout(1);
}
```

i	j
1	1 → n
2	1,2 → n
3	1,2,3 → n
4	1,2,3,4 → n
⋮	⋮ →
n	1,2,....n → n

1+2+3+4+...n

$O(n^2) + O(n)$

$O(n^2)$

$\frac{n(n+1)}{2} = \frac{n^2+n}{2} = \frac{n^2}{2} + \frac{n}{2}$

$= n^2 + n$

$\therefore O(n^2)$

```
n×n pattern:
int n = 4;
for(i=1; i<=n; i++){
  for(j=1; j<=n; j++){
    sout(*);
  }
  sopln();
}
```

i=1	n
i=2	n
i=3	n
i=4	n
⋮	⋮
i=n	n

n×n = $O(n^2)$.

```
for(i=1; i<n; i=i*2){
  sout(i);
}
```

$O(\log_2 n)$

Prime no:

```
for(i=2; i*i<=n; i++){
  if(n%i==0){
    —1—
  }
}
```

n = 36

i = 2, 3, 4, 5, 6

Sout → O(n).

$O(\sqrt{n})$.