Complete Beginner For loop-2

Structure of for loop :

```
for( initialisation ; condition check ; update ) {
    // statement & loop work
}
```

```
int i=1;
for( i) {
    SOP(i);
}
```

↳ Compile time Error.

```
int i=1;
for( ; i<=4 ; i++) {
    SOP(i);
}
```

→

```
for(;;) {
    SOP("hello");
}
```

→ Iterate infinite time

NOTE: We can skip any of three part i.e. initialisation, condition check & update.

important → semicolon (;) must be present.

Problem: First digit , Last digit

↳ Given N, print first digit and last digit of a Number.

Ex N = 7 4 3 2 8

first digit = 7

last digit = 8

Last digit

```
int ld = N%10;
```

```
int fd = 0 ;
```

→ Before loop

[print all digits]

initialisation	condition check	loop work	update
N	N > 0	N%10	N = N/10
74328	true.	fd = 8	7432
7432	true	fd = 2	743
743	true	fd = 3	74
74	true.	fd = 4	7
7	true	fd = 7	0
0	[false] → Break point of loop.		

first digit = fd

value of fd after the loop?

fd = 7

code:

```
int N = scn.nextInt();
int ld = N % 10;
int fd = 0;
for( ; N > 0 ; N = N / 10) {
    fd = N % 10;     OR     fd = N;
}
```

Number of factors:

print all the factors of a number?

Definition of factor: factor of number N will be any number that completely divides N:

N % i == 0 that means 'i' is factor of 'N'

N = 24 → 1, 2, 3, 4, 6, 8, 12, 24

N = 48 → 1, 2, 3, 4, 6, 8, 12, 16, 24, 48

N:
- Smallest factor of any Number = 1 Range of [1, N]
- largest factor = N
- Any number greater than N, can be factor]x.

conclusion

initialise $i = 1;$

condition $i <= N$

update $i = i+1$ OR $i++$

loop work → check if 'i' is factor or not

$N \% i == 0$

i is factor of N.

```
int N = scn.nextInt();
for( int i=1; i<=N; i++) {
    if ( N % i == 0) {
        SOP(i);
    }
}
```

what happend if $i=0$

what happend:

$i = 0$

Problem: Prime Numbers

Definition of prime Number: A number which is divisible by 1 and itself is know as prime Number. ~~incomplete~~

↳ Edge case : $N=1$

N is divisibe by 1

N is divisible by N

1 is Exception

correct definition : A positive number for which count of factor is equal to 2 (i.e. 1 and N itself) is called prime Number.

Smallest prime Number = 2

N	factor Count	factors	Status
factor Count (1) =	1	1	not prime
factor Count (2) =	2	1, 2	prime.
factor Count (11) =	2	1, 11	prime.
factor Count (0) =	Infinitie	N - {0}	prime is define for N > 1

dividend = quotient * divisor + Remainder

N = 0 → Exception

- N % 1 = 0
- N % 2 = 0 — that means 2 is factor
- N % 3 = 0 — " " 3 " "
- N % i = 0 → there is infinite number of factor for '0'.

N / i = 0

i ∈ Every number Except i=0

0 = q * 0 + 0

infinite factor.

Composite Number: Any number which can be created using primes numbers are called composite Number.

N = 10 → 2 * 5 (prime, prime) — Composite Number

N = 12 → 2 x 2 x 3 (prime) — composite Number

7 = 7 * 1 (prime)

Given a number N, print whether it "prime" or "Composite" or "neither prime nor composite".

Approach: check if no. of factr is equal to '2'.
then if it is equal → N → prime otherwise not.

How to find factr: Try to divide every number from 1 to N. if N % i == 0, i is factr
increas count of factr.

N > 0

```
int N = scn.nextInt();
int factorCount = 0;
for (int i = 1; i <= N; i++) {
    if (N % i == 0) {
        factrCount++;
    }
}
if (factorCount == 1) {
    SOP (Neither Prime Nor Composite);
} else if (factrCount == 2) {
    SOP (Prime);
} else {
    SOP (Composite)
}
```

N = 6.

factor count = 0

true → factor count ++

i	i <= N	N % i == 0	i++
1	true.	factorCount = 1	2
2	true.	factor count = 2	3
3	true.	factor count = 3	4
4	true.	——	5
5	true.	——	6
6	true.	factor count = 4	7
7	False] → Break point of loop		

factor count

```
if ( f.c. == 1) {
    SOP ( n. prime. n. comp.) ;
} else if (f.c == 2){
    S.O.P. (prime);

} else {
    S.O.P. (Composite);   ⟶ composite.
}
```

f.c = 4

lets make a dry Run- (Question : check N is prime or not)

N = 24.

i	i <= 24	factor count
1	true.	1
2	true	2
3	true.	3.
4	true.	4.
5	true.	4
6	true.	5
.	.	.
.	.	.
.	.	.

wastage of time.

when factor count > 3 → we have to STOP now

→ Stop here to save our time.

Is there any construct is available in language from which we can stop the loop.

↳ using 'break' we can stop the loop

```
int fc=0;
for(int i=1; i<=N; i++){
    if(N%i==0){
        fc++;
    }
    if(fc==3){
        S.O.P(composite);
        break;
    }
}
if(f.c==1){
    S.O.P(n.p.n.c);
} else if(f.c==2){
    S.O.P(composite);
}
```

break → it will stop just outer loop.

1. → Initialisation. → skip.
2 → condition. → skip ≡ true condition
3 → update → skip.

```
int i=1;
for(1;2;3){
    if(i==5){
        break;
    }
    S.O.P("Hello");
    i++;
}
```

skipped ≡ true.

i=1.

1

2.

3.

4

5.

condition.	Loop work			update.
	if check	printing	increment	
true.	false.	Hello	i=2	
	false	Hello	i=3.	printing Hello 4 times.
	false	Hello	i=4.	
	false	Hello	i=5.	
	true			

↳ loop will stop because of break statement.

Problem: print all the odd number:

Print all the odds numbers from 1 to 10.

```
for(int i=1; i<=10; i=i+2) {
    S.O.P(i);
}
```

```
for(int i=1; i<=10; i++){
    if(i%2==0) {
        // i is even
        continue;
    }
    SOP(i); → print odd numbers
}
```

```
for(int i=1; i<=10; i++){
    if(i%2==1) {
        // i is odd
        continue;
    }
    SOP(i); → print Even
}
```

Problem: Print all Numbers from 1 to 10 Except 2 & 5.

N=10 → 1 3 4 6 7 8 9 10

```
for(int i=1; i<=10; i++){
    if(i==2 && i==5) {
        break;
    }
    SOP(i);
}
```

break Statement is Reachable ??

true → true → unreachable

```
for(int i=1; i<=10; i++) {
    if(i!=2 || i!=5) {
        SOP(i);
    }
}
```

	condition	loop	update
i = 1	true	✓	2
i = 2	true	✓	
i = 3			

Issue is clear ✗

AND

1
2

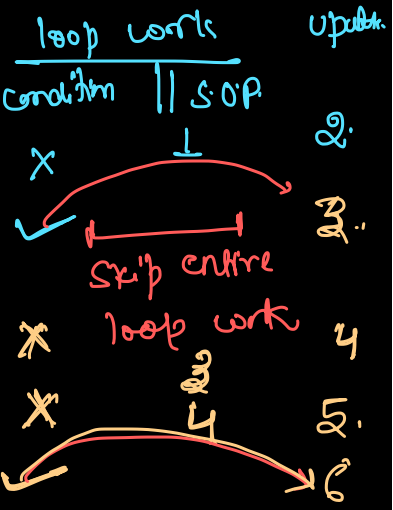
i

construct → continue → skip current iteration. start from the next iteration.

```
for(int i=1; i<=10; i++){
    if(i==2 || i==5){
        continue;
    }
    SOP(i);
}
```

skip iteration

i	i<=10 condition
1	true
2	true
3	true
4	true
5	true



NOTE: If "continue" is encountered skip entire loop work for current iteration & move for next iteration.

Doubt Session.

break → stop just outer loop. continue.

```
for(  ) {
    for( ; ; ) {
        if( true ) {
            if( true ) {
                break;
            }
            Statement 1
            Statement 2
        }
    }
}
```

Stop just outer loop.

Ex

Statement

```
for( ; ; ){
    if( ){
        continue;
    }
    Statement 1
    Statement 2
}
```

Stop Entire loop work

move to the next iteration

Print all the primes for 1 to N.

```
for( int n = 1;  n <= N ; n++) {
    Number = n  ⟶  check if 'i' is prime or NOT
        { put the logic to check if n is
          prime or not.
}
```

convert double into ceil number & floor numbe ?

type casting

double n = 8.77

ceil (8.77) = (int) (8.77 + 1) = 9

floor (8.77) = 8

n+1 = 0
↳ n → as ceil

```
int f = (int)n;  // floor
(8.97+1) = 9   8.97+1 == 0        8.97
if ( int (n+1) - (n+1) == 0) {    8.00
     9   8+1 == 0
     n is in Integral form.
    ceil = n;
} else {
    ceil = (int) (n+1);
}
```

(8.97+1) = 9 //