FUNCTIONS

FACTORIAL

N = 5

ans : 120

$$n! = n \times (n-1) \times (n-2) \times \ldots \times 1$$

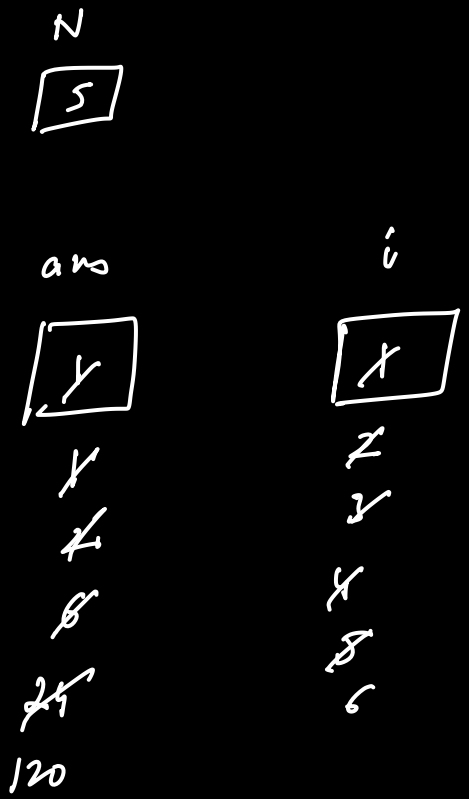
$= 1 \times 2 \times 3 \text{ ----- } \times n$

$3! = 3 \times 2 \times 1 = 6$

$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

```
int N;
cin >> N;

int ans = 1;
for(int i=1 ; i<= N; i++)
{
        ans *= i;
}
cout << ans;
```



BINOMIAL COEFFICIENT

$$^{n}C_{r} = \frac{n!}{r!(n-r)!}$$

Input $n = 5$

$r = 2$

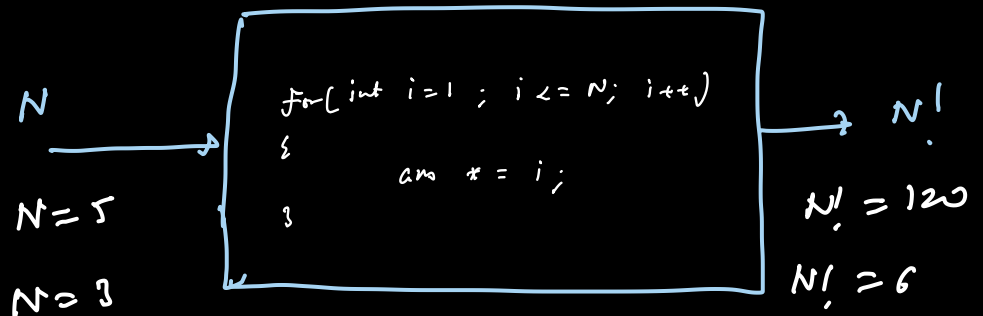
$$^{5}C_{2} = \frac{5!}{2! \times 3!} = \frac{120}{2 \times 6} = 10$$

i/p : n and r

o/p : $^{n}Cr = \frac{n!}{r!\,(n-r)!}$

```
int  N, R;

cin >> N >> R;
```




```
int  nFact = 1;

for(int i=1 ; i<=N; i++)
{
    nFact *= i;
}
```

N!

```
int rFact = 1;
for(int i=1; i<=R; i++)
{
    rFact *= i;
}
```

R!

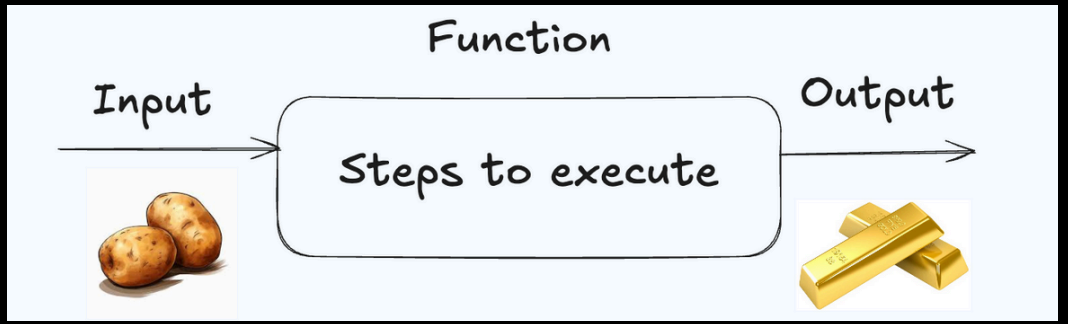
```
int nrFact = 1;
for(int i=1; i<=N-R; i++)
{
    nrFact *= i;
}
```

(N-R)!

```
cout << nFact / (rFact * nrFact);
```

FUNCTIONS

- Set of statements
- Write only once
- Reuse it multiple times
- Makes code:
  - Cleaner
  - Shorter
  - Easier to debug



FUNCTION SYNTAX

```
return_type function_name(parameters)
{
    // function body
    return value;
}
```

- Return type → what the function gives back
- Function name
- Parameters → inputs
- Return statement

```
int factorial ( int N)
{
    int ans = 1;
    for(i=1; i<=N; i++)
    {
        ans *= i;
    }
    return ans;
}
```

EXAMPLES

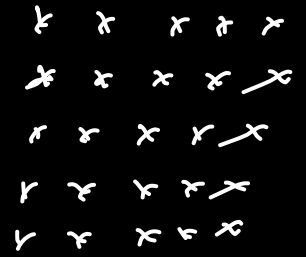
```
int sum2(int a, int b)
{
    return a + b;
}
```

```
int sum3(int a, int b, int c)
{
    return a + b + c;
}
```

```
int factorial(int n)
{
    int ans = 1;
    for(int i = 1; i <= n; i++)
    {
        ans *= i;
    }
    return ans;
}
```

EXAMPLES

```
void print1toN(int n)
{
    for(int i = 1; i <= n; i++)
    {
        cout << i << endl;
    }
}
```

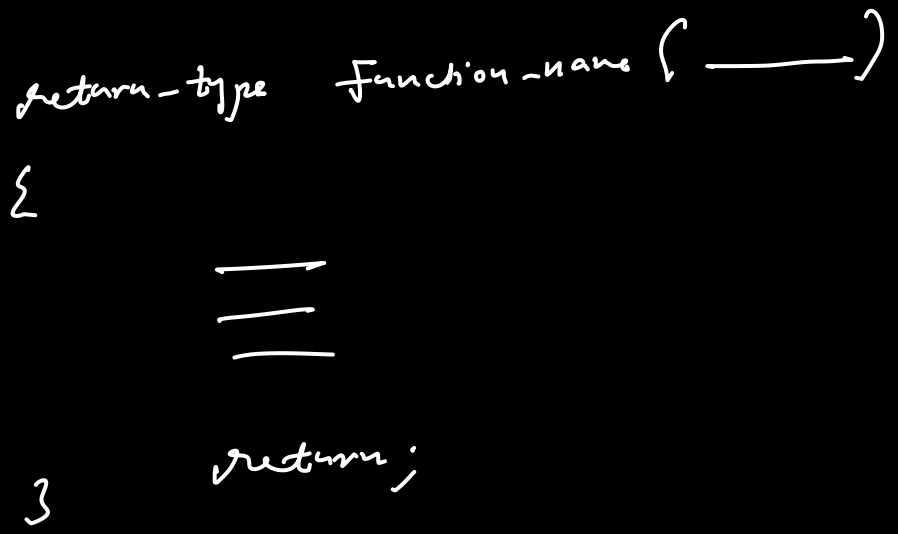


```
void printSquare(int n, char ch)
{
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
        {
            cout << ch;
        }

        cout << endl;
    }
}
```

FUNCTION RULES

- Number of Parameters Must Match
- Return Type Must Match
- Return Ends the Function
- A Function May or May Not Return
  - int, double, bool → returns value
  - void → prints only



Factors

12 → 1, 2, 3, 4, 6, 12 (6 factors)

15 → 1, 3, 5, 15 (4 factors)

Smallest Factor

N → 1

Largest Factor

N

```
For( i = 1;  i<=N;  i++)
{
        if ( i is a factor of N)
        {
            cout << i << " ";
        }
}
```

N % i == 0

Divide N with i, remainder = 0

1

2

i

1 2 3 4 5 6 7 8 9 10 11 12

3
4
6
12

1) Print Factors

2) Count Factors

Prime Numbers ⟶ Exactly 2 factors → (1), (N)

1 is not a prime number

1 2 3 5 7 11 13 17 19 23

[1] [1,2] [1,3] [1,5] [1,7] [1,11] ---

(N) → Prime or not

(N) → Print all prime numbers from 1 to N

isPrime(i)

N = 10

2 3 5 7

N = 20

2 3 5 7 11 13 17 19

```
for( i = 1; i <= N; i++)
{
    if( i is Prime)
    {
        cout << i << " ";
    }
}
```