

Recursion - 1

Lecture-27

Raghav Garg

COLLEGE
WALLAH

What and Why?

Recursion \rightarrow recurrence relation

What? \rightarrow • Loop ki replacement

• problem = something + subproblem

\rightarrow function calling itself

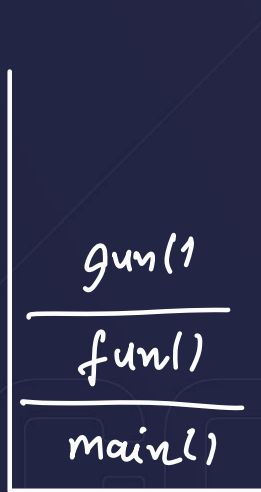
Why \rightarrow To solve problems

Function calls

```
✓ #include<iostream>
✓ using namespace std;
✓ void gun(){
  ✓ return; // khatam
  cout<<"Hello CW"<<endl;
}
✓ void fun(){
  ✓ cout<<"Hello PW"<<endl;
  ✓ gun();
  ✓ return;
}
✓ int main(){
  ✓ fun();
}
```

• Hello PW

0



Call stack

Function calling itself

```
#include<iostream>
using namespace std;
void fun(){
    cout<<"Hello PW"<<endl;
    fun();
}
✓int main(){
    ✓fun();
}
```

- Hello PW
- Hello PW
- Hello PW

COLLEGE
WALLAH

Function calling itself

```
✓ #include <iostream>
✓ using namespace std;
void fun(int n){
    if(n==0) return;
    cout<<"Hello PW"<<endl;
    fun(n-1);
}
✓ int main(){
    ✓ fun(3);
}
```

```
void fun(int 0 n){
    ✓ if(n==0) ✓ return;
    cout<<"Hello PW"<<endl;
    fun(n-1);
}
```

```
void fun(int 1 n){
    ✓ if(n==0) return;
    ✓ cout<<"Hello PW"<<endl;
    ✓ fun(n-1);
}
```

```
void fun(int 3 n){
    ✓ if(n==0) return;
    ✓ cout<<"Hello PW"<<endl;
    ✓ fun(n-1);
}
```

```
void fun(int 2 n){
    ✓ if(n==0) return;
    ✓ cout<<"Hello PW"<<endl;
    ✓ fun(n-1);
}
```

Output

- Hello PW
- Hello PW
- Hello PW
-

Function calling itself

Classwork : Print goodmorning 'n' no. of times, where 'n' is user input.

Ques : Make a function which calculates the factorial of n using recursion.

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

$$= 5 \times 4!$$

$$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$$

$$= 7 \times 6!$$

$$n! = n \times n-1 \times n-2 \times n-3 \times \dots \times 3 \times 2 \times 1$$

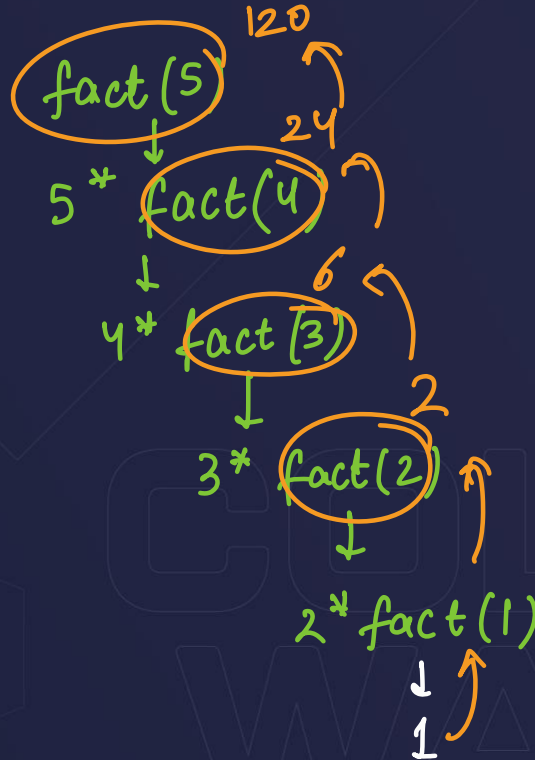
$$n! = n \times (n-1)!$$

factorial of n =
1 to n numbers ka
product

COLLEGE
WALLAH

Ques : Make a function which calculates the factorial of n using recursion.

```
#include<iostream>
using namespace std;
int fact(int n){
    return n*fact(n-1);
}
int main(){
    cout<<fact(5);
}
```



Ques : Make a function which calculates the factorial of n using recursion

```
int main(){
    cout<<fact(5);
}
```

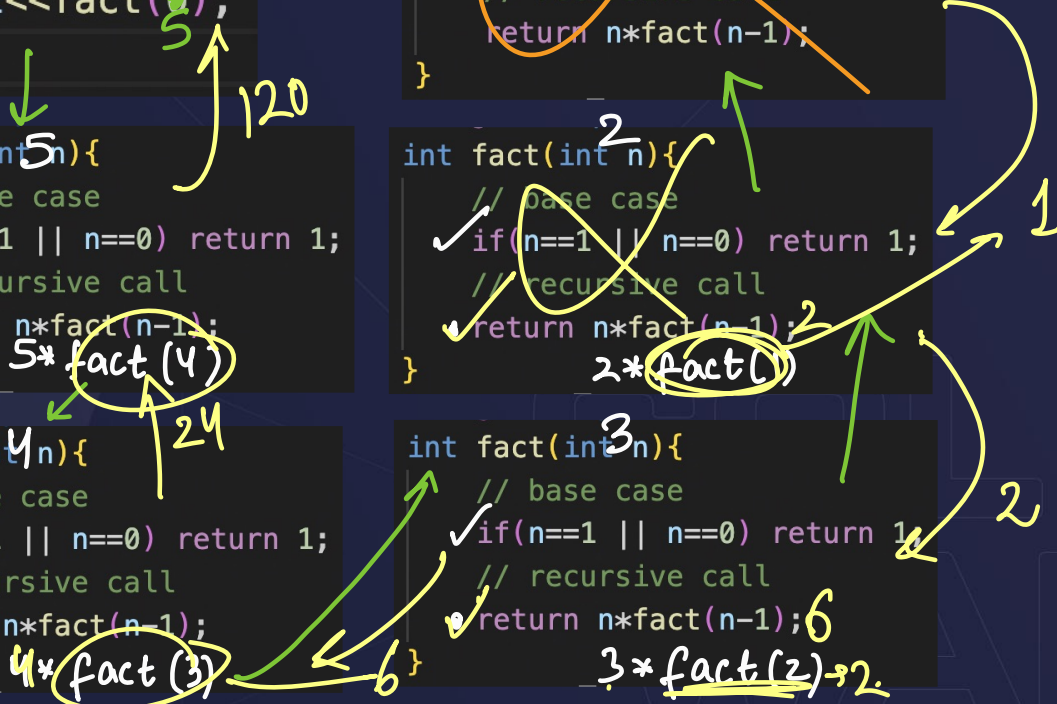
```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```

```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```

```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```

```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```

```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```



Ques : Make a function which calculates the factorial of n using

```
int main(){
    cout<<fact(4);
}
```

```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```

```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```

```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```

```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```

```
int fact(int n){
    // base case
    if(n==1 || n==0) return 1;
    // recursive call
    return n*fact(n-1);
}
```

$n! \rightarrow 2^n$ ops

↓

T.C. = $O(n)$

S.C. = $O(n)$ stack frames

→ $O(n)$

Ques : Print n to 1

```
int n;  
cin >> n;  
print(n);
```

Output

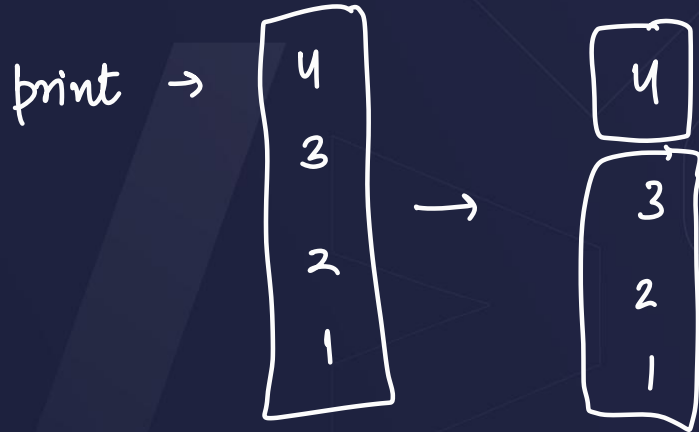
$n = 4$

4

3

2

1



Ques : Print 1 to n (extra parameter)

```
void print(int i, int n){
    if(i > n) return;
    cout << i << endl;
    print(i+1, n);
}
```

Handwritten annotations: '1' above 'i', '5' above 'n', a green checkmark on the first line, and a large green 'X' over the entire function.

```
void print(int i, int n){
    if(i > n) return;
    cout << i << endl;
    print(i+1, n);
}
```

Handwritten annotations: '2' above 'i', '5' above 'n', green checkmarks on the first two lines, and a large green 'X' over the entire function.

```
void print(int i, int n){
    if(i > n) return;
    cout << i << endl;
    print(i+1, n);
}
```

Handwritten annotations: '5' above 'i', '5' above 'n', green checkmarks on the first three lines, and a large green 'X' over the entire function.

```
void print(int i, int n){
    if(i > n) return;
    cout << i << endl;
    print(i+1, n);
}
```

Handwritten annotations: '4' above 'i', '5' above 'n', green checkmarks on the first three lines, and a large green 'X' over the entire function.

```
void print(int i, int n){
    if(i > n) return;
    cout << i << endl;
    print(i+1, n);
}
```

Handwritten annotations: '3' above 'i', '5' above 'n', green checkmarks on the first three lines, and a large green 'X' over the entire function.

```
void print(int i, int n){
    if(i > n) return;
    cout << i << endl;
    print(i+1, n);
}
```

Handwritten annotations: '6' above 'i', '5' above 'n', a green checkmark on the first line, a green box around the 'return;' statement, and a large green 'X' over the entire function.

output

- 1
- 2
- 3
- 4
- 5
-

Ques : Print 1 to n (after recursive call)

```
int main(){
    print(4);
}
```

```
void print(int n){
    if(n==0) return; // base case
    print(n-1); // call
    cout<<n<<endl; // kaam
}
```

```
void print(int n){
    if(n==0) return; // base case
    print(n-1); // call
    cout<<n<<endl; // kaam
}
```

```
void print(int n){
    if(n==0) return; // base case
    print(n-1); // call
    cout<<n<<endl; // kaam
}
```

```
void print(int n){
    if(n==0) return; // base case
    print(n-1); // call
    cout<<n<<endl; // kaam
}
```

```
void print(int n){
    if(n==0) return; // base case
    print(n-1); // call
    cout<<n<<endl; // kaam
}
```

Output

• 1

• 2

• 3

• 4

•

Recursive Function

Base Case

Kaam

Call

Kaam

return

COLLEGE
WALLAH

Ques : Print sum from 1 to n (Parameterised)

0

```
void sum(int sum, int n){  
    if(n == 0) return;  
    sum(sum+n, n-1);  
}
```

COLLEGE
WALLAH

Ques : Print sum from 1 to n (Parameterised)

```
void sum1toN(int sum, int n){
    ✓ if(n=0){
        cout<<sum<<endl;
        return;
    }
    ✓ sum1toN(sum+n, n-1);
}
```

Handwritten annotations: 0, 4, 4, 3, and a large 'X' over the recursive call.

```
void sum1toN(int sum, int n){
    ✓ if(n=0){
        cout<<sum<<endl;
        return;
    }
    ✓ sum1toN(sum+n, n-1);
}
```

Handwritten annotations: 9, 1, 10, 0, and a large 'X' over the recursive call.

```
void sum1toN(int sum, int n){
    ✓ if(n=0){
        ✓ cout<<sum<<endl;
        ✓ return;
    }
    sum1toN(sum+n, n-1);
}
```

Handwritten annotations: 10, 0, 'yes', and a large 'X' over the recursive call.

```
void sum1toN(int sum, int n){
    ✓ if(n=0){
        cout<<sum<<endl;
        return;
    }
    ✓ sum1toN(sum+n, n-1);
}
```

Handwritten annotations: 4, 3, 7, 2, and a large 'X' over the recursive call.

```
void sum1toN(int sum, int n){
    ✓ if(n=0){
        cout<<sum<<endl;
        return;
    }
    ✓ sum1toN(sum+n, n-1);
}
```

Handwritten annotations: 7, 2, 9, 1, and a large 'X' over the recursive call.

.10

Ques : Print sum from 1 to n (Return type)

```
int sum(int n){
    if(n==0) return 0;
    return n + sum(n-1);
}
```

```
int fact(int n){
    if(n==1 || n==0) return 1;
    return n * fact(n-1);
}
```

$sum(n) = n + sum(n-1);$ $fact(n) = n * fact(n-1)$

$$1 \text{ to } 6 \rightarrow 6 + 1 = 6 + \boxed{5 + 4 + 3 + 2 + 1} = sum(6)$$

$$6 + sum(5) = sum(6)$$

Ques : Make a function which calculates 'a' raised to the power 'b' using recursion.

Input \rightarrow a, b

cout << pow(a, b);

$$a^b = a^1 \times a^{b-1}$$

$$\text{power}(a, b) = a^1 \times \text{power}(a, b-1)$$

$$a^b = \underbrace{a \times a \times a \times a \times a \dots}_{b \text{ times}}$$

$$2^4 = \underbrace{2 \times 2 \times 2 \times 2}_{4 \text{ times}}$$

Diagram illustrating the recursive steps for calculating 3^5 using the formula $3^n = 3 * 3^{n-1}$:

3^5
 \downarrow
 $3 * 3^4$
 \downarrow
 $3 * 3^3$
 \downarrow
 $3 * 3^2$
 \downarrow
 $3 * 3^1$
 \downarrow
 $3 * 3^0$ (base case)

```
return = ax power(a, b-1);
```

Calculating Time and Space complexity

Recursive Solution

```
int pow(int a, int b){  
    if(b==0) return 1;  
    return a*pow(a,b-1);  
}
```

$$T.C. = O(b)$$

$$S.C. = O(b) \text{ stack frames}$$

Iterative Solution

```
int power(int a, int b){  
    int p = 1;  
    for(int i=1;i<=b;i++){  
        p *= a;  
    }  
    return p;  
}
```

$$T.C. = O(b)$$

$$S.C. = O(1)$$

Calculating Time and Space complexity

```
int pow(int3 a, int4 b){
    if(b==0) return 1;
    return a*pow(a, b-1);
}
```

```
int pow(int3 a, int3 b){
    if(b==0) return 1;
    return a*pow(a, b-1);
}
```

```
int pow(int3 a, int0 b){
    if(b==0) return 1;
    return a*pow(a, b-1);
}
```

```
int pow(int3 a, int1 b){
    if(b==0) return 1;
    return a*pow(a, b-1);
}
```

```
int pow(int3 a, int2 b){
    if(b==0) return 1;
    return a*pow(a, b-1);
}
```

$2^*b \rightarrow$
space
khatel

Next Lecture

Fibonacci Series
↑

Understanding multiple calls in Recursion

More problems on Recursion

| | | | | | | | | | | | |
|---|---|---|---|---|---|----|----|----|----|----|-----|
| 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 | ... |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | |

COLLEGE
WALLAH