You might get some idea here:

<http://www.cplusplus.com/articles/D2N36Up4/>

<https://www.youtube.com/watch?v=4agL-MQq05E>

<https://www.youtube.com/watch?v=ozmE8G6YKww> (case in java but the same concept)

<https://www.youtube.com/watch?v=NvVYd08NUXI>

Here are some simple recursive functions provided below. Try to trace the output of each functions.

**Function – 1:**

void p(int x)

{

if (x == 0) cout << "the end" << endl;

else{

cout << x << ", ";

p(x - 1);

}

}

**Calling statement: p(10);**

**Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Function – 2:**

int factorial(int x)

{

if (x == 1)

{

cout << "multiply by " << x << endl;

return 1;

}

else{

cout << "multiply by " << x << endl;

return x \* factorial(x - 1);

}

}

**Calling statement:** cout << "The result of 3! is " << factorial(7) << endl;

**Output:**

**Function – 3:**

void func1(int x)

{

if (x <= 5)

{

func1(x + 2);

cout << x << ", ";

func1(x + 1);

}

}

**Calling statement:** func1(2);

**Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Function – 4:**

int fib(int x) //Fibonacci number;

{

if (x <= 2) return x;

else{

return fib(x - 2) + fib(x - 1);

}

}

**Calling statement:** cout << fib(5) << endl;

**Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Function – 5:**

void printB(int x)

{

if (x != 0)

{

cout << x << ", ";

printB(x - 1);

cout << x << ", ";

}

}

**Calling statement:** printB(5);

**Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Function – 6:**

int rec(int n)

{

if (n == 0) return 1;

else return 3 \* rec(n - 1);

}

**Calling statement:** cout << rec(3) << endl;

**Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**