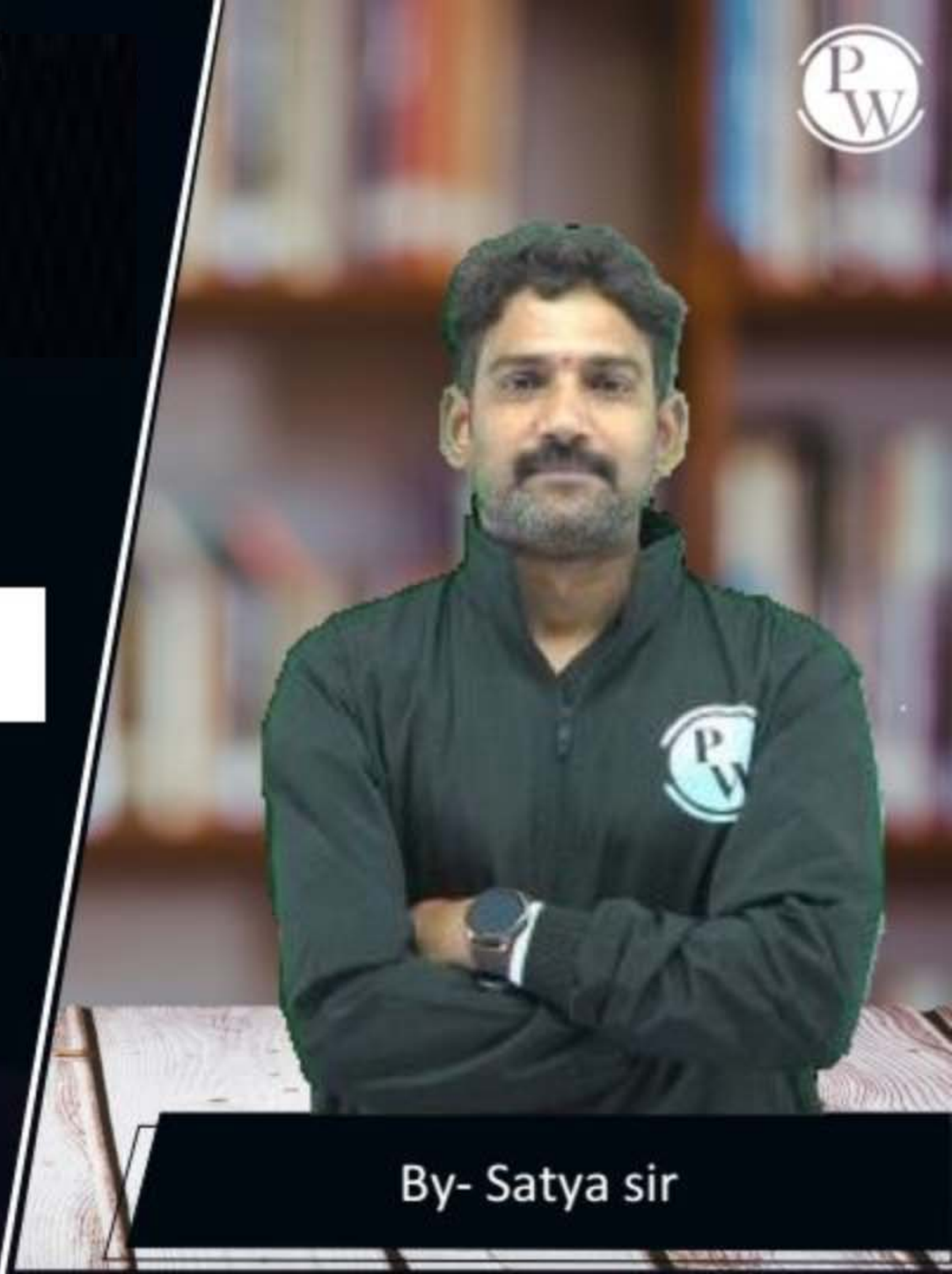
CS & IT ENGINEERING

'C' Programming

Functions

Lecture No.- 03



By- Satya sir

Recap of Previous Lecture

- Recursion
  - The Process of Calling a function by itself.
- Types of Recursion
  - Direct : Head Recursion, Tail Recursion, Tree Recursion, Nested Recursion
  - Indirect
- Call-by-Value Vs Call-by-Reference
  (formal args gets Effected) (Actual args gets Effected)

Topics to be Covered

- Types of Recursion
  - Head
  - Tail
  - Tree
  - Nested

  } Direct

- Indirect

Topic : Recursion - 2

Head Recursion : If after calling a function recursively if more body [statement(s)] Exist to Execute.

Syntax :

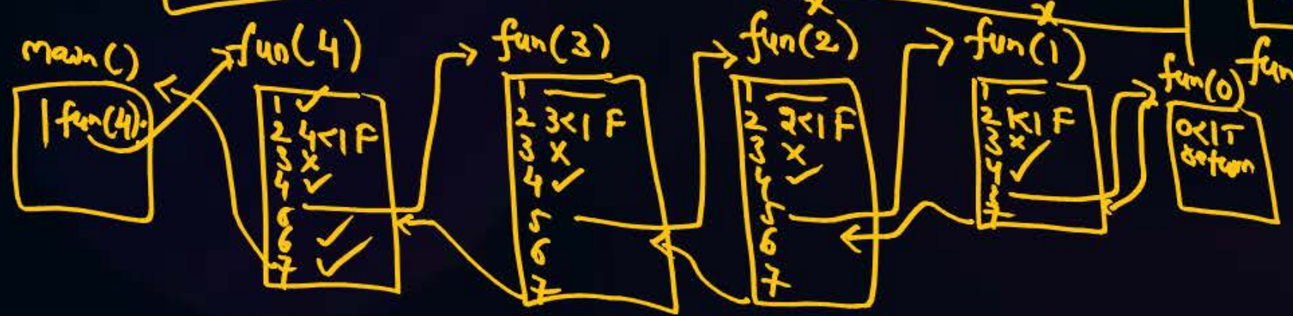
```
Return type function ( argument (s))
{
    // Base Case

    // Recursive Case

    Stmt ;
    Stmt ;
    :
}
```

Ex:

```
1.  Void fun (int x)
    {
2.      if (x<1)          // Base Case
3.          return ;
4.      else
        {
5.          fun( x-1);    // Recursive Case
6.          Printf(" Hi");
7.          Printf("%d", x);   } statements after recursive Calling
        }
    }

    Void main( )
    {
        fun(4);
    }
```

o/p: Hi 1 Hi 2 Hi 3 Hi 4

Function Call Stack

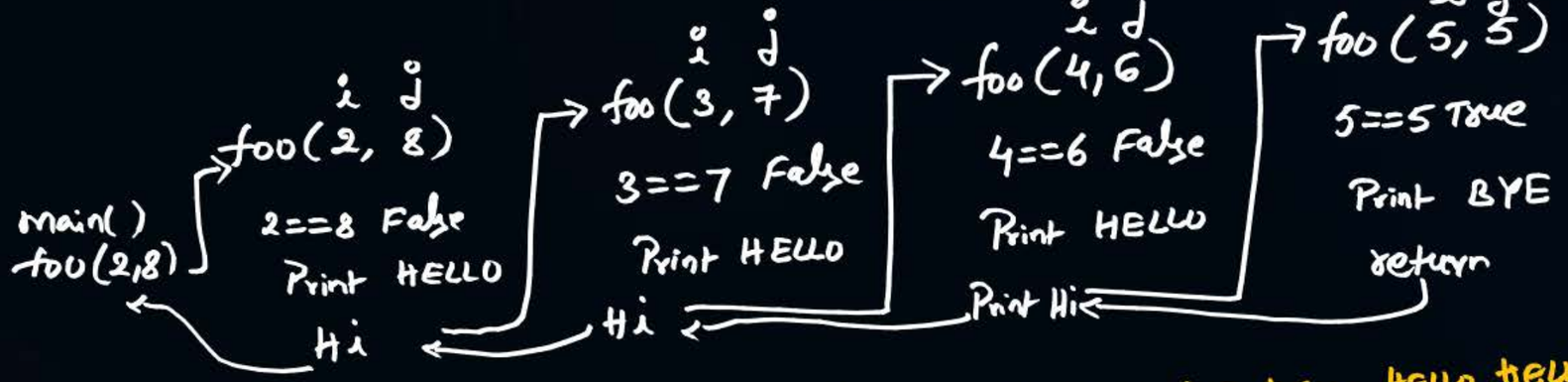


Topic : Recursion - 2

o/p: HELLO HELLO HELLO BYE Hi Hi Hi

Ex:1

```
void foo(int i, int j)
{
    if(i==j)
    {
        Printf("BYE ");
        return;
    }
    else
    {
        Printf("HELLO");
        foo(i+1, j-1);
        Printf("Hi ");
    }
}
→ void main()
{
    foo(2,8);
}
```



a) HELLO Hi Hi Hi BYE HELLO HELLO HELLO

b) HELLO HELLO HELLO BYE Hi Hi Hi ✓

c) HELLO HELLO Hi Hi BYE

d) HELLO HELLO HELLO Hi Hi Hi

Topic : Recursion - 2

$X(8,3) = 8+3-\underset{11}{X(7,4)}+\underset{36}{X(6,6)}$

$= 11-11+36 = 36$

Ex:2

```
int X(int i, int j)
{
    if(i == j)
        return (i*j);
    else if(i<j)
        return (i+j);
    else
        return (i+j-X(i-1, j+1) + X(i-2, j+3));
}
Void main( )
{
    Printf("%d", X(10,0));
}
```

o/p: 22

$X(6,5)$

$6+5-\underset{11}{X(5,6)}+\underset{12}{X(4,8)}$

$= 11-11+12 = 12$

$X(7,4)$

$7+4-\underset{12}{X(6,5)}+\underset{12}{X(5,7)}$

$= 11$

$X(10,0)$

$10+0-X(9,1)+X(8,3)$

$= 10+0-24+36 = 22$

$X(9,1)$

$9+1-X(8,2)+X(7,4)$

$= 10-(-3)+11 = 24$

$X(8,2)$

$8+2-X(7,3)+X(6,5) = 10-25+12$

$= -3$

$X(7,3)$

$7+3-X(6,4)+\underset{11}{X(5,6)} = 10-(-4)+11$

$= 25$

$X(6,4)$

$6+4-\underset{25}{X(5,5)}+\underset{11}{X(4,7)} = -4$

Topic : Recursion - 2

Tail Recursion = If Recursive Calling is the Last statement of function body, Then it is Said to be Tail Recursion.

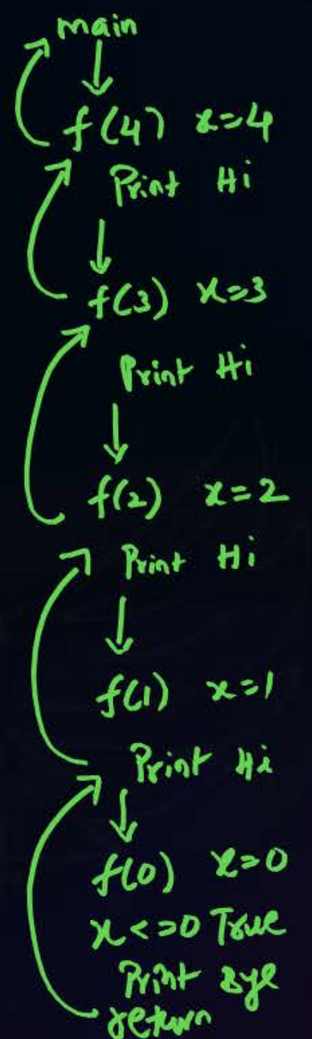
Syntax:

```
Returntype xName(argument(s))
{
    // Base Case

    // Stmt ;
    // Stmt ;
       :
       :
    // Stmt ;
    // Name(arg); // Recursive Calling
}
```

Ex:

```
Void f(int x)
{
  if(x<=0)
  { Printf("Bye ");
    return;
  }
  else { Printf("Hi");
         f(x-1);
       }
}

 Void main( )
 { f(4);
 }
```

o/p: Hi Hi Hi Hi Bye



Topic : Recursion - 2

Ex:

```
int f(int i, int j)
{
    if ( i < j)
        return (i+j)
    else if ( i > j )
        return (i-j)
    else
        return f(i+2, j-3);
}
void main( )
{
  Printf(" %d", f(1,7));
}
```

$f(1,7)$ $i=1$, $j=7$

$1<7 \Rightarrow$ return $(1+7)=8$

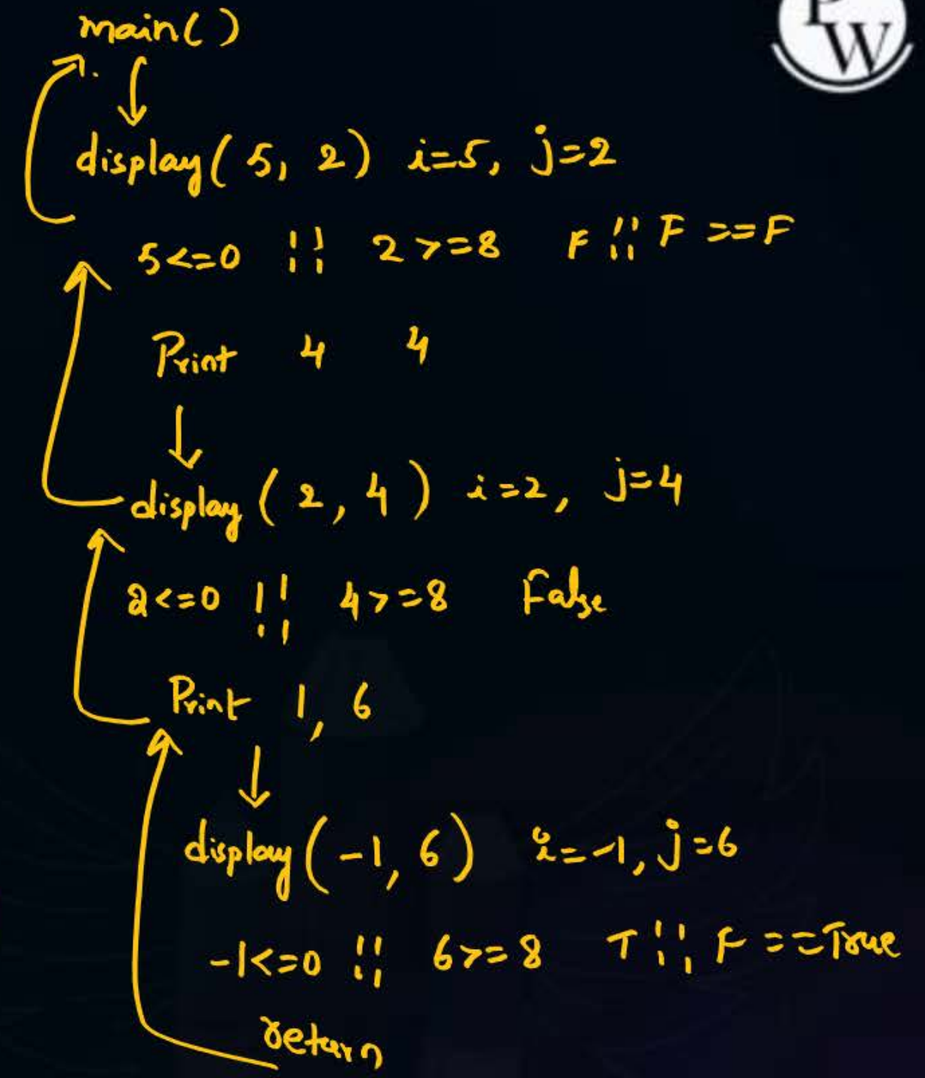
8

Topic : Recursion - 2

Ex:

```
Void display(int i, int j)
{
    if((i<=0) || (j>=8))
        return;
    Printf("%d  %d \n", (i-1),(j+2))
    display(i-3, j+2);
}
→ Void main( )
{
    display(5, 2);
}
```

o/p: 4 4
1 6



Topic : Recursion - 2

Tree Recursion

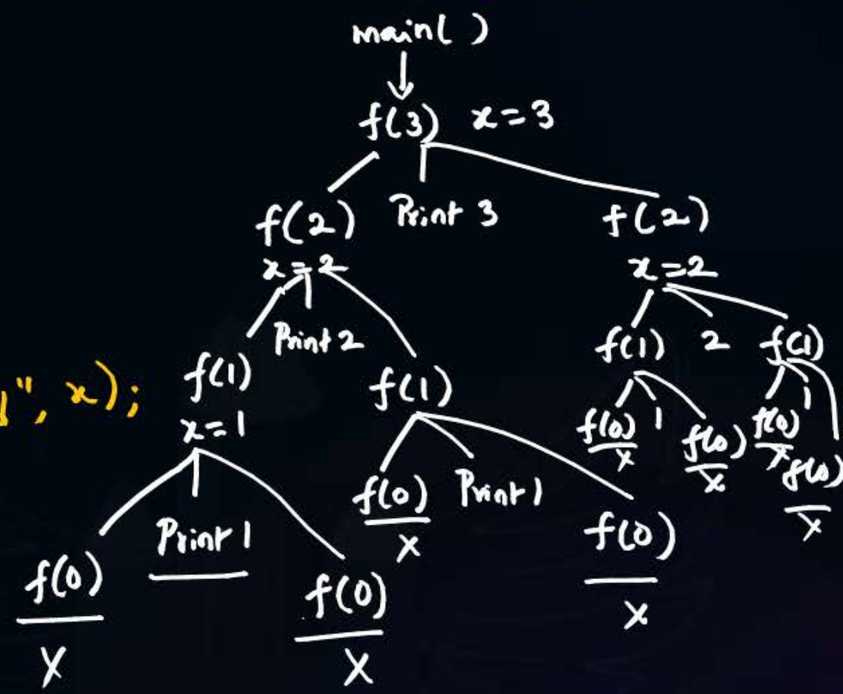
If, In a Recursive function, recursive calling is made multiple times, Then it is said to be Tree Recursion.

Syntax:

```
Return type  Name (Argument(s))
{
    // Base Case

    // Recursive Case
      Name (arg);
      Name (arg);
        Stmt;
      Name (arg);
        :
}
```

Ex:

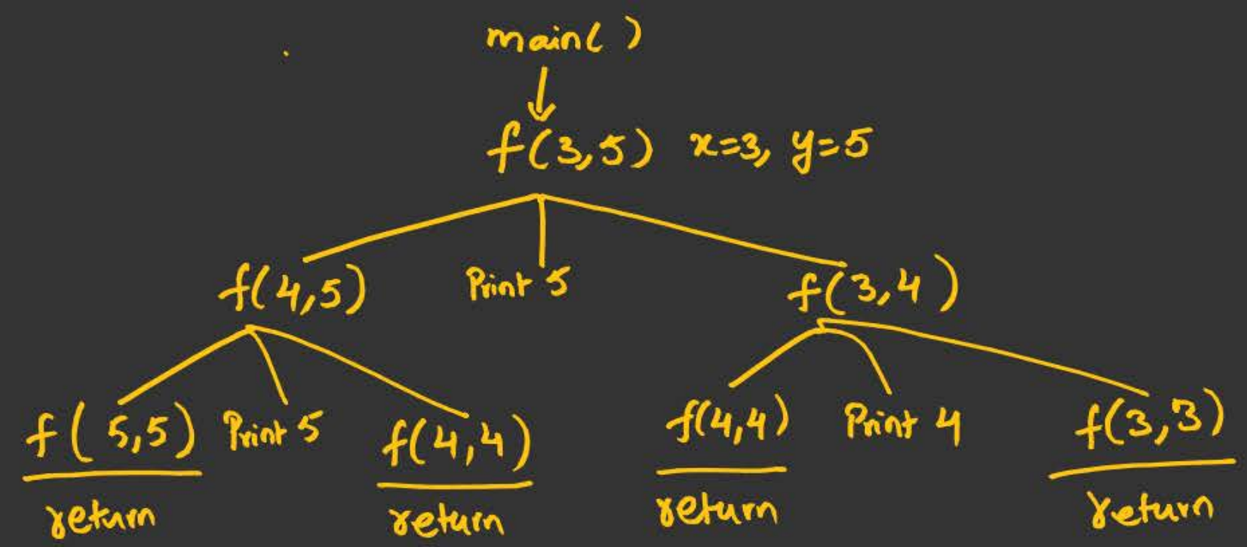
```
void f(int x)
{  if(x<=0)
     return;
   else
   {  f(x-1);
      Printf("%d", x);
      f(x-1);
   }
}
void main()
{
   f(3);
}
```



O/P: 1 2 1 3 1 2 1

Ex:-1

```
Void main( )
{
    f(3,5);
}
Void f(int x, int y)
{
    if (x == y) return;
    else if(x<y)
       { f(x+1, y);
         Printf("%d", y);
         f(x, y-1);
       }
    else
       { f(x-1, y);
         Printf("%d", x);
    } } f(x, y+1);
```



o/p: 5 5 4

Nested Recursion : While a Recursive call, if another recursive call is made, Then it is said to be Nested Recursion.

Syntax:

```
ReturnType Name (arguments)
{
    // Base Case

    // Recursive Case
        Stmt;
        :
        Name (Name (Name(arg_)));
}
```

Ex:

```
int x(int i)
{
  if(i<=1)
    return i;
  else
  {  Printf("%d",(i+1));
     return x(x(i-1));
  }
}
void main( )
{  x(3);
}
```

o/p: 4 3

main( )
↓
x(3) i=3
Print 4 ✓
↓
x(x(2))
x(1)
returns 1

x(2) i=2
↓
Print 3 ✓
↓
x(x(1))
x(1)
returns 1

x(1) i=1
return 1

Indirect Recursion : A function is called itself, through some other function(s).

Ex:

```
A( )
{
   B( );
}
B( )
{
   C( );
}
C( )
{
   A( );
}
```

Example

```
void A(int x)
{  if(x<=1) return;
   else
   { Printf("Hi");
     B(x+1);
   }
}
void B(int y)
{  if(y>=5) return;
   else
   { Printf("Hello");
     C(y*2);
   }
}
```

```
Void C(int z)
{  if(z%5==0) return;
   else
   { Printf("Bye");
     A(z/2);
   }
}
Void main( )
{  A(3);
}
```

o/p: Hi Hello Bye Hi

main( )
↓
A(3) x=3
↓
B(4) y=4
↓
C(8) z=8
↓
A(4) x=4
↓
B(5) y=5
5>=5 True
return.

2 mins Summary

Types of Recursion

- Direct
  - Head
  - Tail
  - Tree
  - Nested
- Indirect

THANK - YOU