

Largest Element in Array

Ex: $n = 5$
 $arr[] = \langle 1, 8, 90, 7, 5 \rangle$

o/p: 90 ✓

Ex: $n = 7$
 $arr[] = \langle 1, 2, 0, 5, 3, 2, 4 \rangle$

o/p: 5 ✓

Ex: $n = 1$
 $arr[] = \langle 5 \rangle$

o/p: 5

First Approach

I/k: $n = 5$
 $arr[] = \langle 1, 8, 90, 7, 5 \rangle$

~~$n = arr[i]$~~

\checkmark for (int $i = 0$; $i < n$; $i++$)
<
 ~~ans~~ = max (ans , $arr[i]$) ;
 8 , 90
>

cout << ans << endl ;
90

int ans = INT_MIN ;
 \hookrightarrow ⊖ve

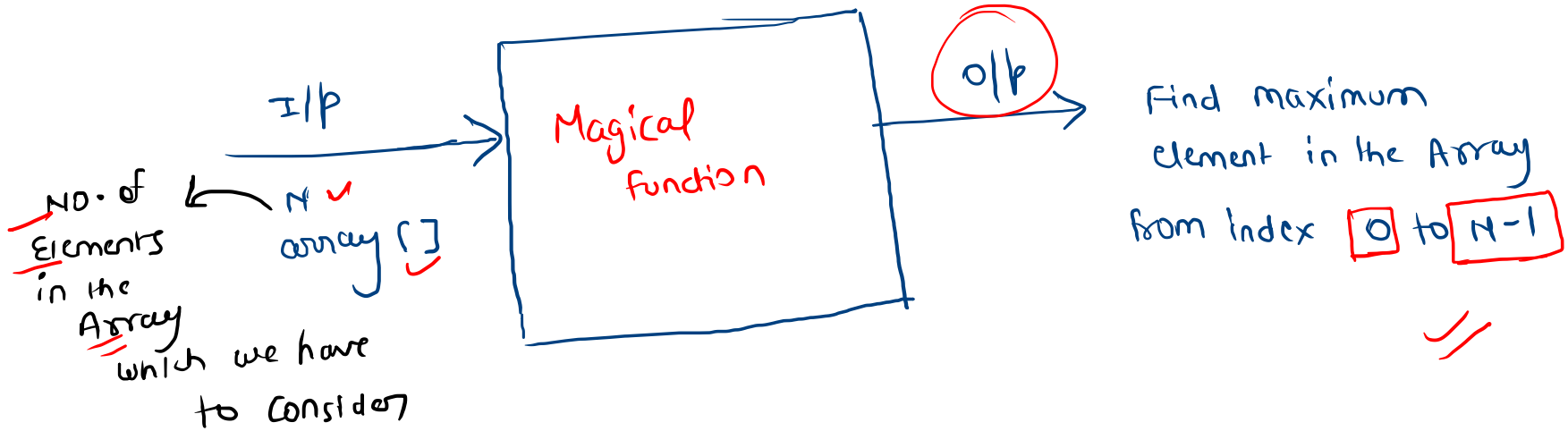
$ans = \cancel{1} \cancel{8} \underline{90}$

Recursive
Solution

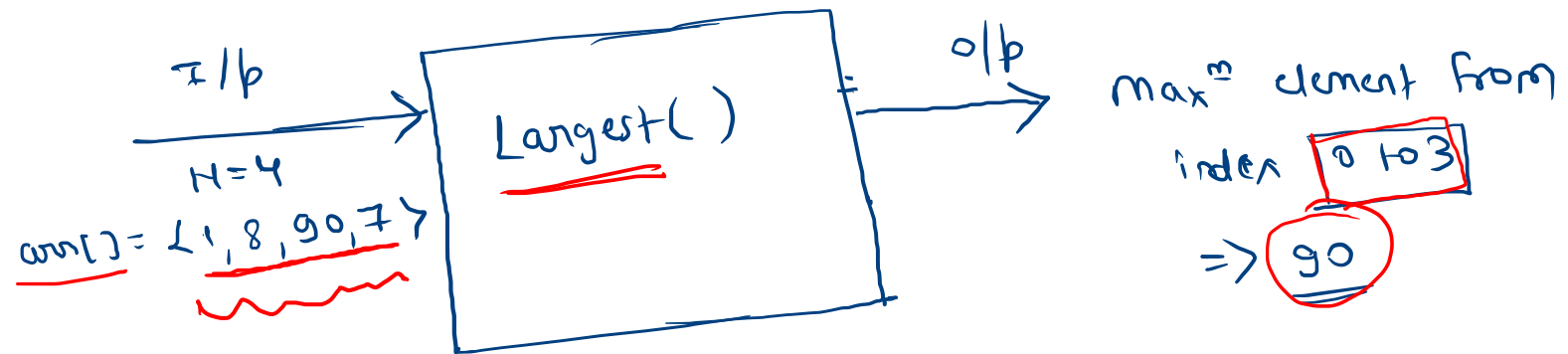
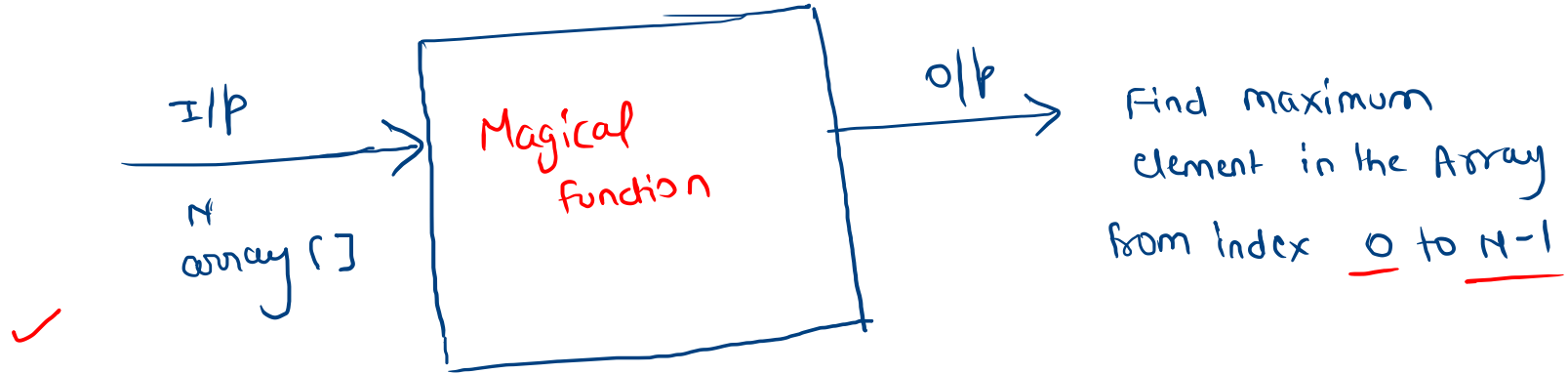
I/k: $n = 5$
 $arr[] = \langle 1, 8, 90, 7, 5 \rangle$

Recursive Solution

I/p: $n = 5$
arr[] = { 1, 8, 90, 7, 5 }
index → 0 1 2 3 4

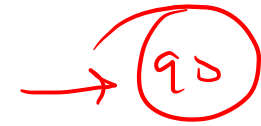
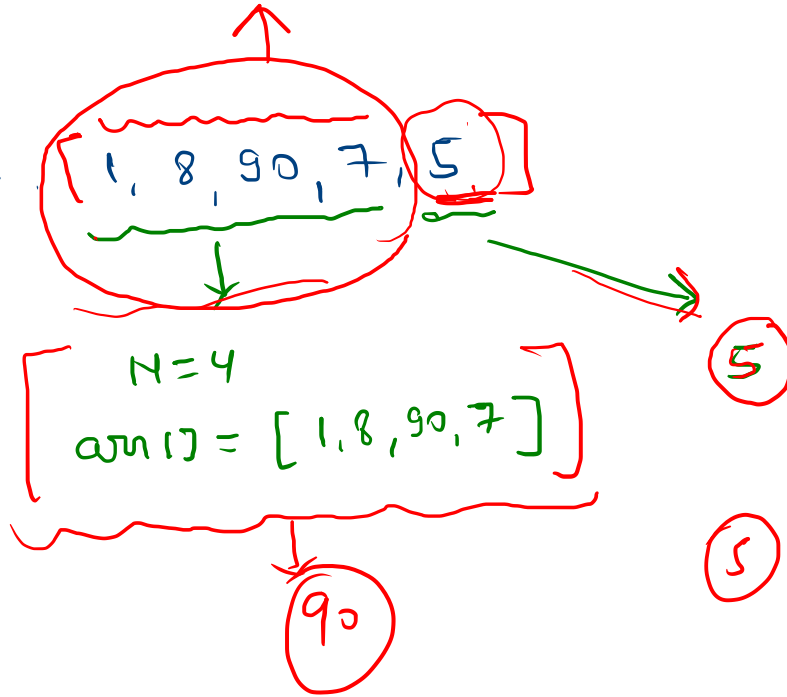


Recursive
Solution



Logical
port

I/k: $n = 5$
 $arr[] =$

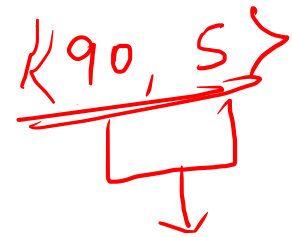
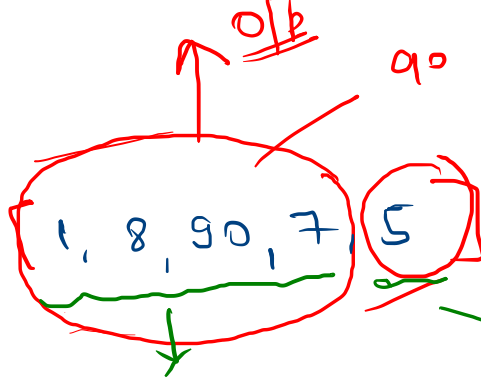


Logical part

I/k:

$n = 5$

$arr[] =$



$N = 4$

$arr[] = [1, 8, 90, 7]$

Magical function



obj ele

5



5

Compare and return output

Logical part

I/p:

$n = 5$

$arr[] =$



Element

$n = 5$

$n-1 = 4$

↓

largest (arr, $n-1$)

o/p

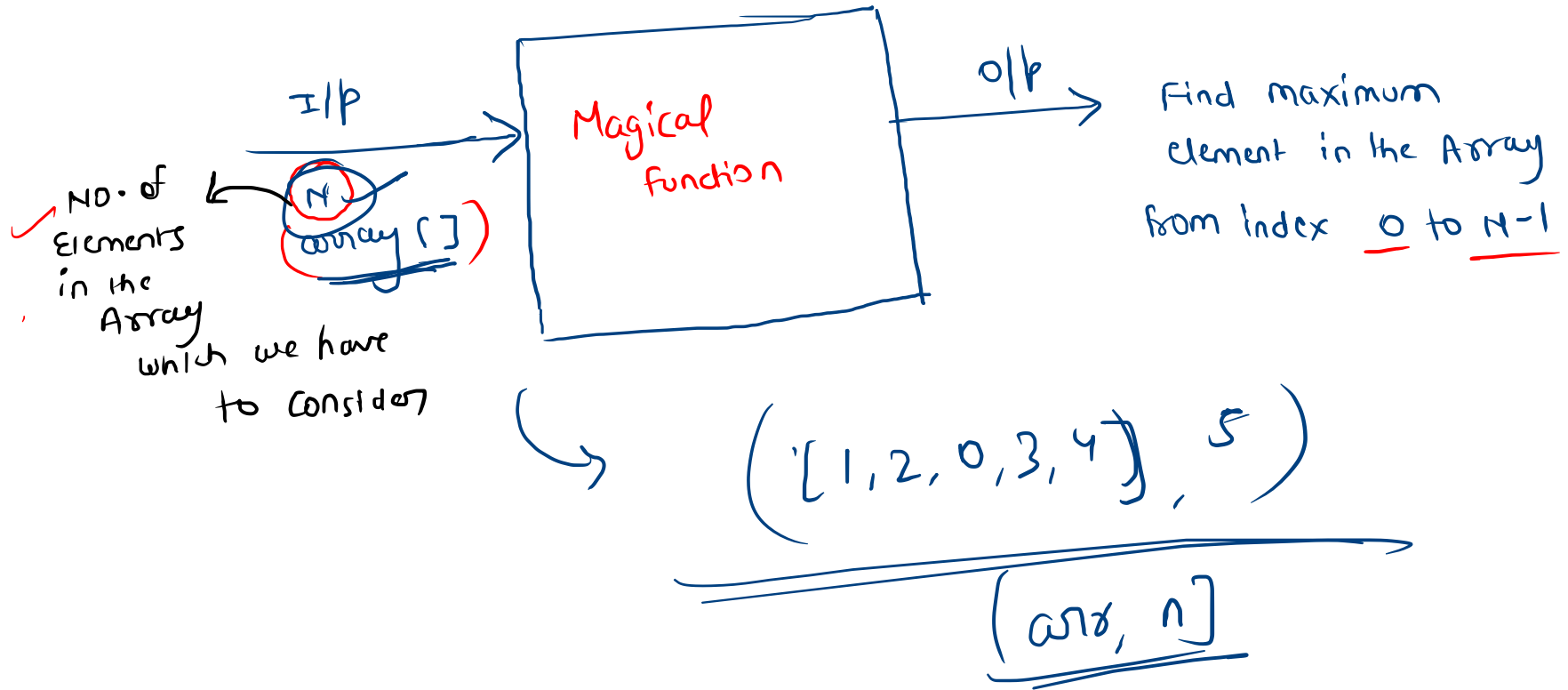
arr [$n-1$]

index

last
elem

largest ([1, 2, 0, 3, 4], 3)

2



Recursive
Solution

```
int Largest ( int arr[], int N )  
{  
    // Base Condition  
    > return max ( Largest ( arr, N-1 ), arr[N-1] );  
}
```

Recursive
Solution

```
int Largest ( int arr[], int H )
{
    // Base Condition
    return max ( Largest ( arr, H-1 ), arr[H-1] );
}
```

Diagram illustrating the recursive call structure:

The diagram shows a hand-drawn array representation `[]`. An arrow points from the space between the brackets to the text "Number of elements". Another arrow points from the closing bracket `]` to the text "index of elements".

Base Condition

Smallest Valid Input \rightarrow check output

int Largest (int arr[], int H)

↓
Variable

$n = 0, 1, 2, 3, \dots$
 x ↓ ↓

Smallest Valid Input

```
if (n == 1) return arr[0];
```

$arr = [5]$
 $n = 1$ \rightarrow $arr[0]$

Recursive
Solution

```
int Largest ( int arr[], int N )  
{  
    ✓ if ( n == 1 ) return arr[0];  
    ✓ return max ( Largest ( arr, N-1 ), arr[N-1] );  
}
```

max

Logical doubt

↓
we are passing
whole array

target (int arr[] , int n)



target ([1, 2, 0, 5] 4, 3)

Diagram illustrating the function call:

- The array `[1, 2, 0, 5]` is circled in red, with an arrow pointing to a circled `5`.
- The value `4` is circled in red, with an arrow pointing to the word itself.
- The value `3` is circled in red, with an arrow pointing to a circled `0 to 3`, which is labeled index.