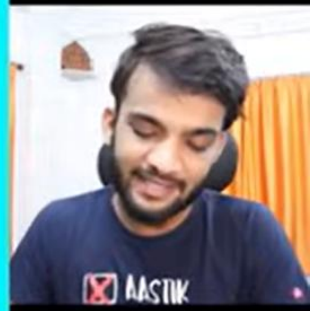
Basic Recursion Problems

- Print Name 5 times
- Print Linearly from 1 to N
- Print from N to 1.
- Print Linearly from 1 to N (But By Backtracking)
- Print from N to 1 (By Backtracking)

TUF

Basic Recursion Problems

- Print Name 5 times
- Print Linearly from 1 to N
- Print from N to 1.
- Print Linearly from 1 to N (But By Backtracking)
- Print from N to 1 (By Backtracking)



TUF

Basic Recursion Problems

- Print Name 5 times
- Print Linearly from 1 to N
- Print from N to 1.
- Print Linearly from 1 to N (But By Backtracking)
- Print from N to 1 (By Backtracking)

TUF

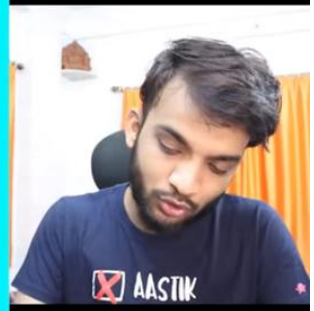

AASTIK
TUF

Print Name N times

TUF

Q Print Name N times using Recursion

```
main()
{
  int n;
  cin >> n;
  f(0, n)
```



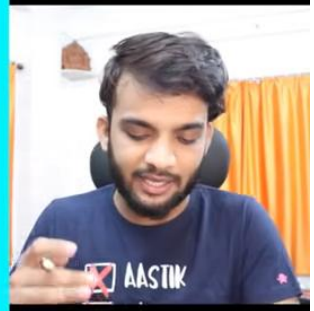
TUF

L1. Implement TRIE | INSERT | SEARCH | STARTSWITH

Q Print Name N times using Recursion

```
void f(i, n)
{
  if (i > n)
      return;
```

```
main()
{
  int n;
  cin >> n;
  f(1, n);
}
```



TUF

Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > n)
        return;
}
```

Base Condition

```
main()
{
  int n;
  cin >> n;
  f(1, n);
}
```

TUF

Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > n)
        return;
    print("raj");
    f(i+1, N);
}
main()
{
    int n;
    cin >> n;
    f(1, n);
}
```

Base condition

TUF

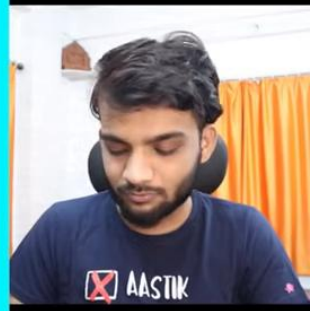
Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > n)
        return;
    print("raj");
    f(i+1, N);
}
main()
{
    int n;
    cin >> n;
    f(1, n);
}
```

Base condition

n = 3

Output

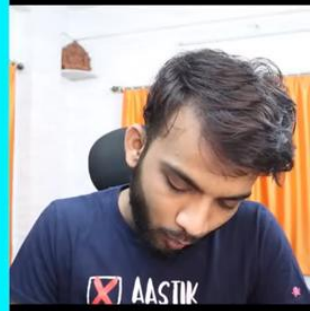


TUF

```
}
int n;
cin >> n;
f(1, n);
}
```

n = 3

f(1,



TUF

(By Backtracking)

Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > 3)
        return;
    print("raj");
    f(i+1, n);
}
main()
{
    int n;
    cin >> n;
    f(1, n);
}
```

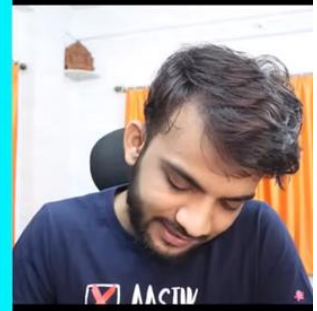
Base condition

f(2,3)
{
...

n = 3

Output

raj



TUF

(By Backstreet)

Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > 3)
        return;
    print("raj");
    f(i+1, n);
}
main()
{
    int n;
    cin >> n;
    f(1, n);
}
```

Base condition

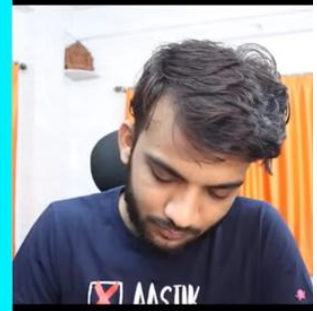
f(2,3)

```
{
    if (i > n)
        return;
    print("ra
```

n = 3

Output

raj



TUF

(By Brute force)

Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > 3)
        return;
    print("raj");
    f(i+1, N);
}

main()
{
}

int n;
cin >> n;
f(1, n);
```

Base condition

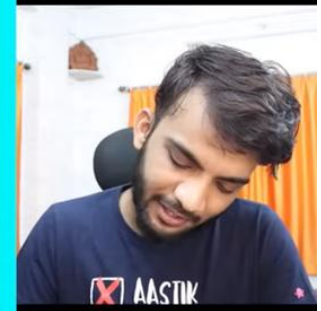
n = 3

f(2,3)

```
{
    if (2 > 3)
        return;
    print("raj")
    f(i+1, N);
}
```

Output

raj



TUF

(By Backstreet)

Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > 3)  X
        return;
    print("raj");
    f(i+1, N);
}

main()
{
    int n;
    cin >> n;
    f(1, n);
}
```

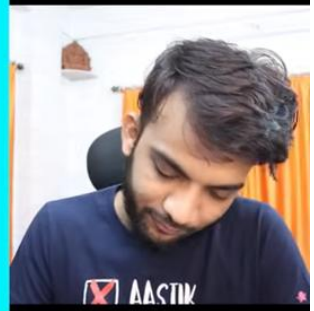
Base condition

n = 3

```
f(2,3)
{
    if (2>3)  X
        return;
    print("raj")
    f(i+1, N);
}
```

Output

raj
raj



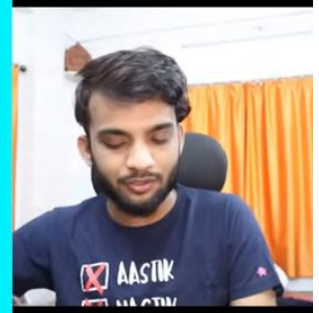
TUF

```
int n;
cin >> n;
f(1, n);
}
```

n = 3

f(1, 3)

f(2, 3)



TUF

(By Backtracking)

Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > 3)  X
        return;
    print("raj");
    f(i+1, N);
}
main()
{
    int n;
    cin >> n;
    f(1, n);
}
```

Base Condition

n = 3

f(2,3)

```
{
    if (2>3)  X
        return;
    print("raj")
    f(i+1, N);
}
```

f(3,3)

Output

raj
raj

(By Backtracking)

Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > n) return;
    print("raj");
    f(i+1, n);
}
main()
{
    int n;
    cin >> n;
    f(1, n);
}
```

Base condition

n = 3

f(2,3)

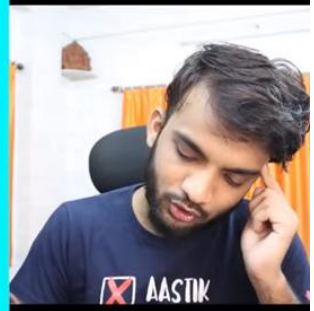
```
{
    if (2>3) return;
    print("raj");
    f(i+1, N);
}
```

f(3,3)

```
{
    if (i>n) return;
    print("raj");
    f(i+1, N);
}
```

Output

raj
raj
raj



TUF

Q Print Name N times using Recursion

```
void f(i, n)
{
    if (i > n) return;
    print("raj");
    f(i+1, N);
}
main()
{
    int n;
    cin >> n;
    f(1, n);
}
```

Base condition

n = 3

f(2,3)

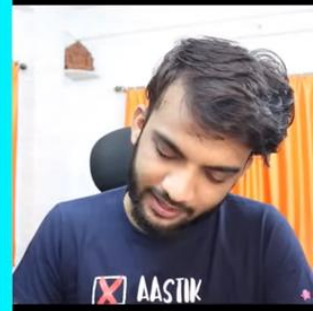
```
{
    if (2>3) return;
    print("raj")
    f(i+1, N);
}
```

f(3,3)

```
{
    if (i>n) return
    print("raj")
    f(i+1, N);
}
```

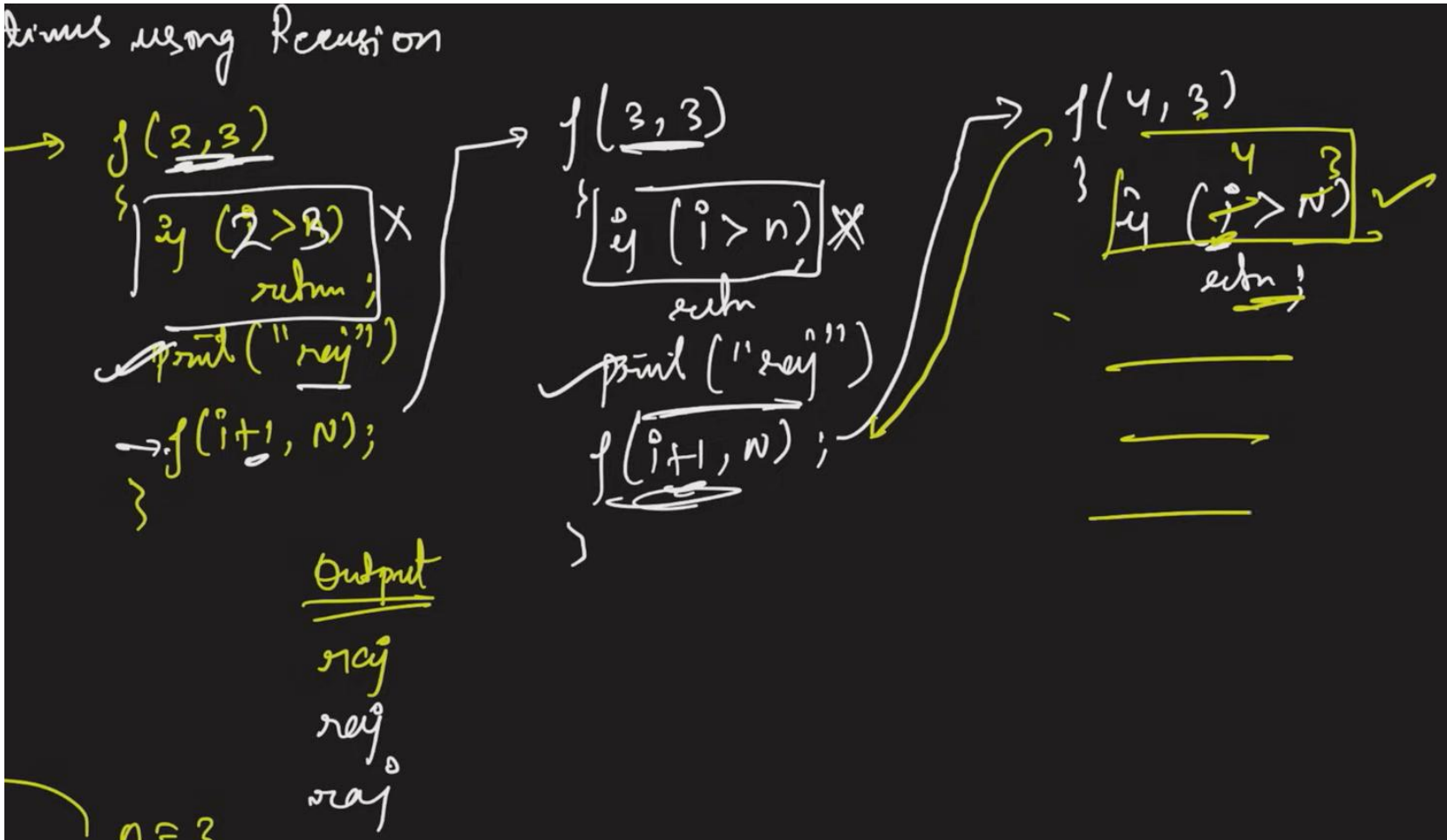
Output

raj
raj
raj



TUF

Prints using Recursion



Output

raj
raj
raj

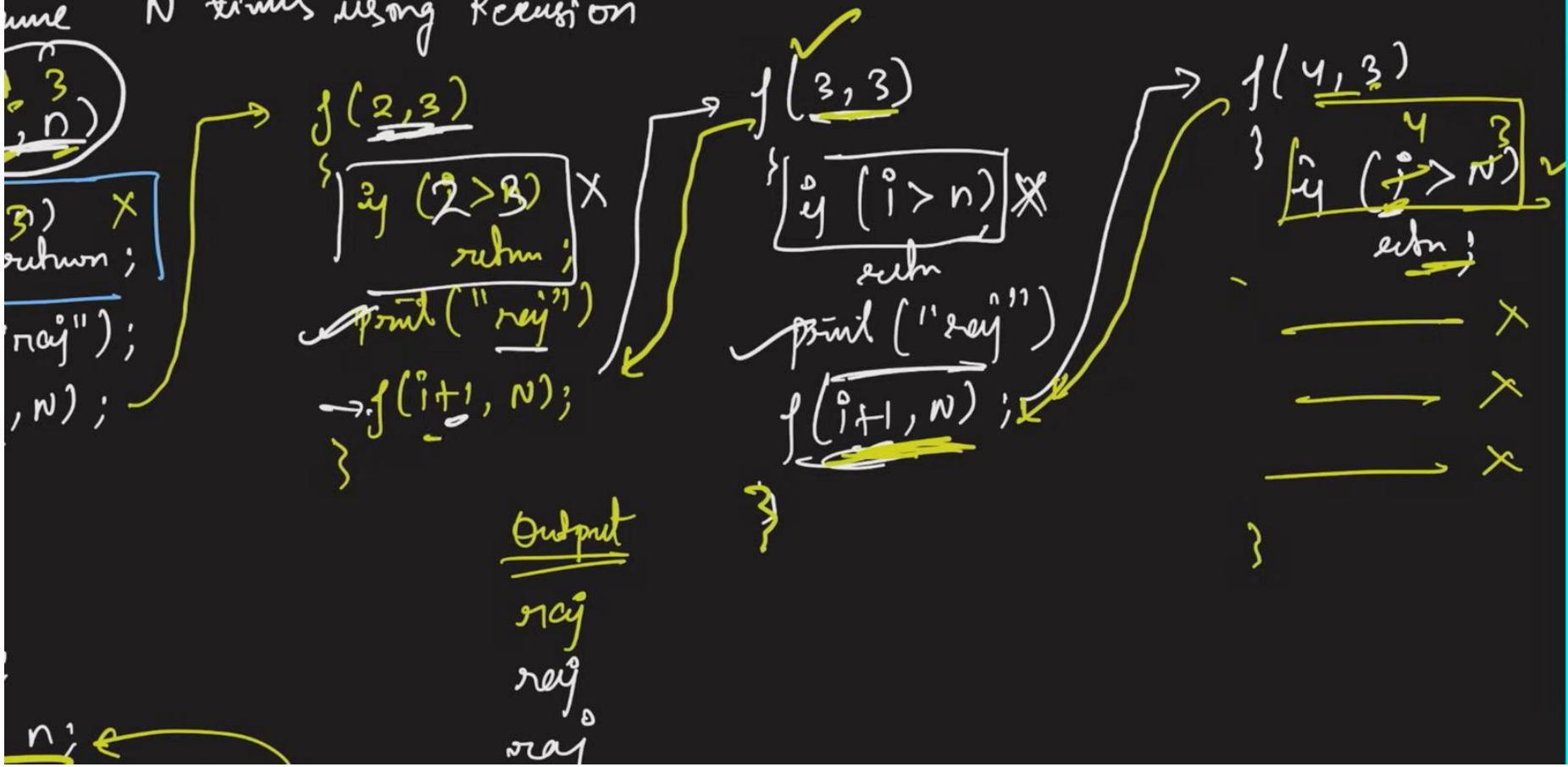
n = 3

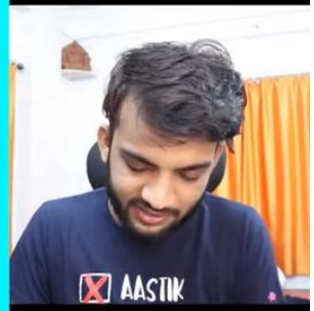
f(1, 3)

TUF

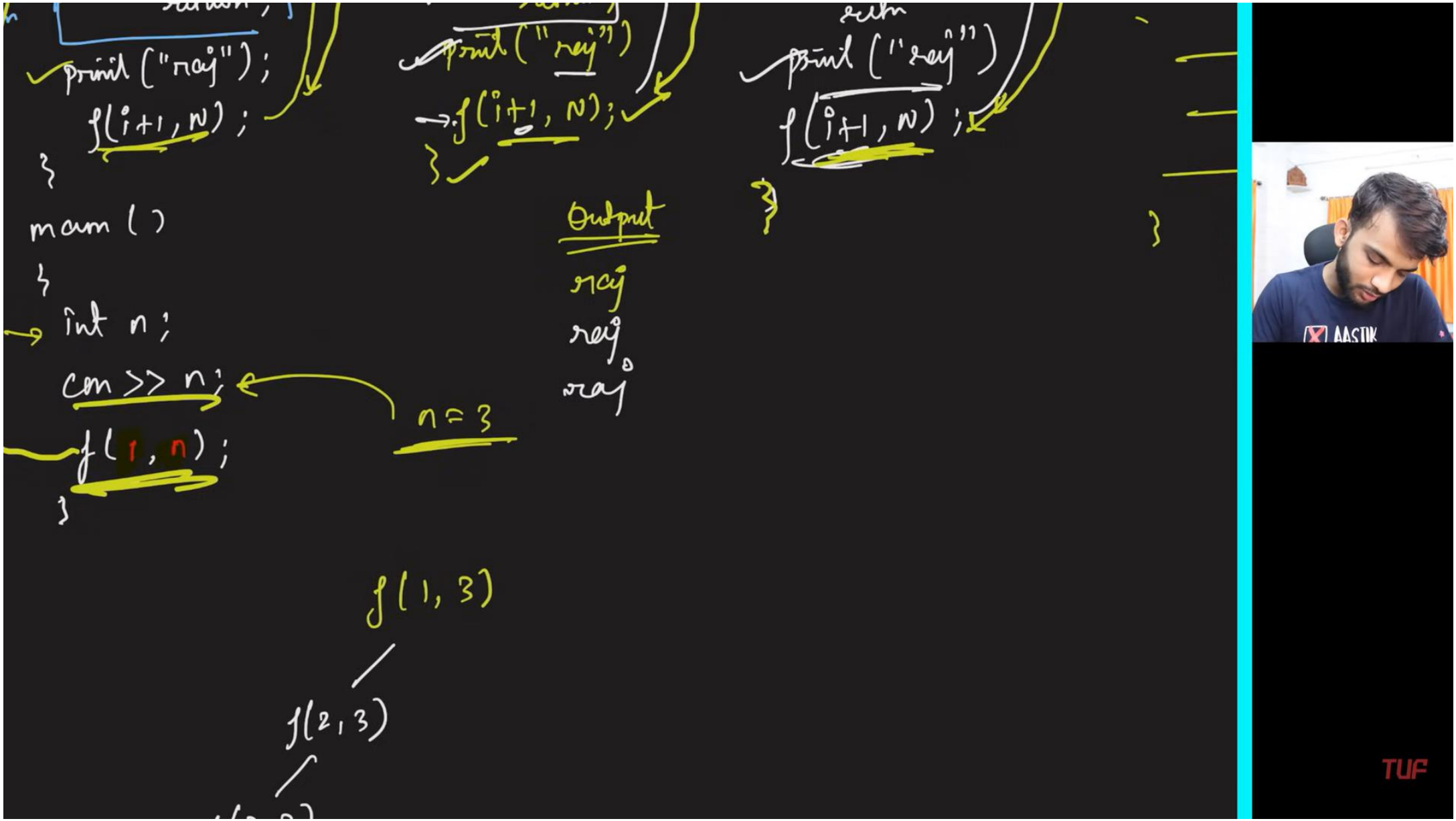
→ Print from N to 1
(By Backtracking)

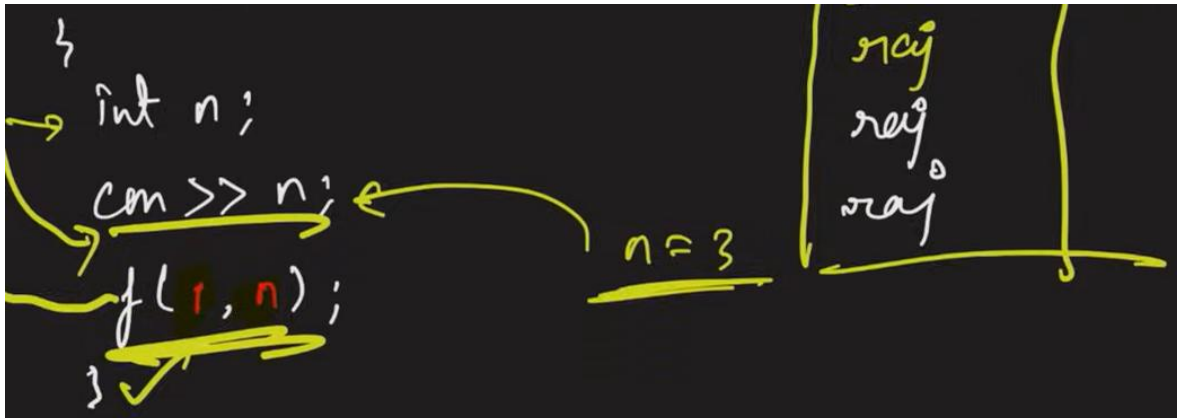
N times using Recursion

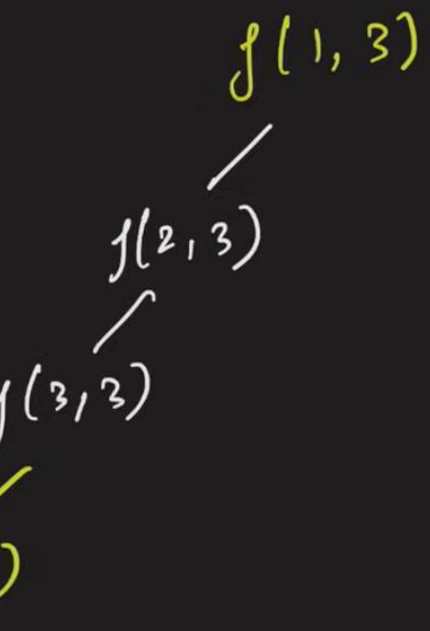


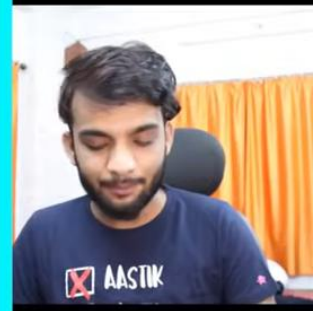


TUF

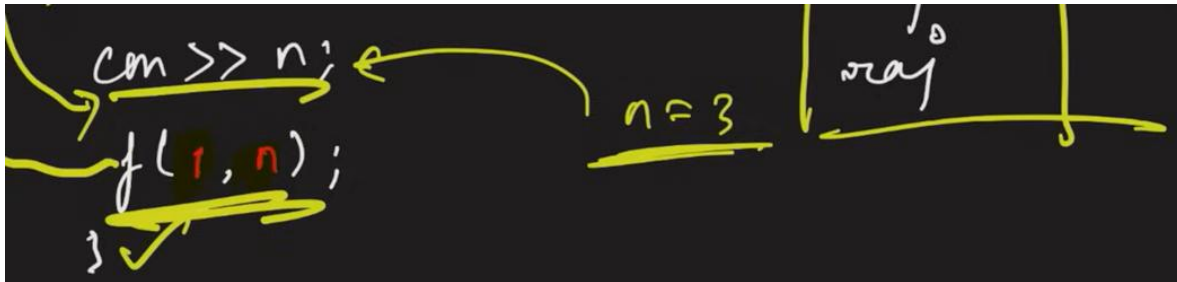

print ("raj");
f(i+1, N);
}
main()
{
int n;
cin >> n;
f(1, n);
}
print ("raj")
f(i+1, N);
}
print ("raj")
f(i+1, N);
}
Output
raj
raj
raj
n = 3
f(1, 3)
f(2, 3)

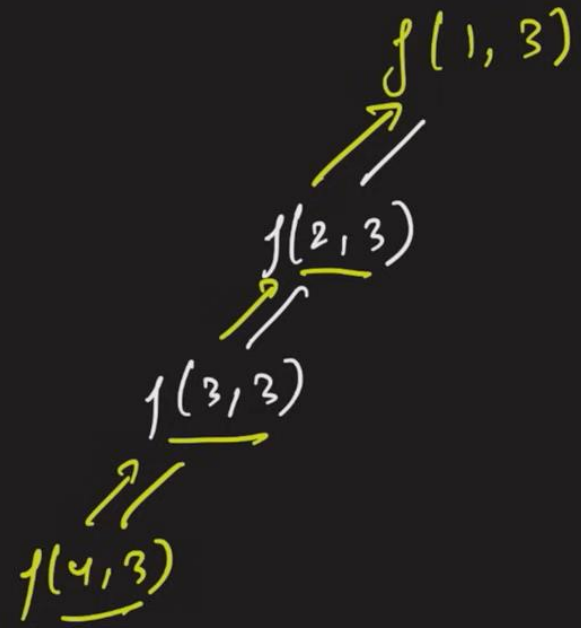




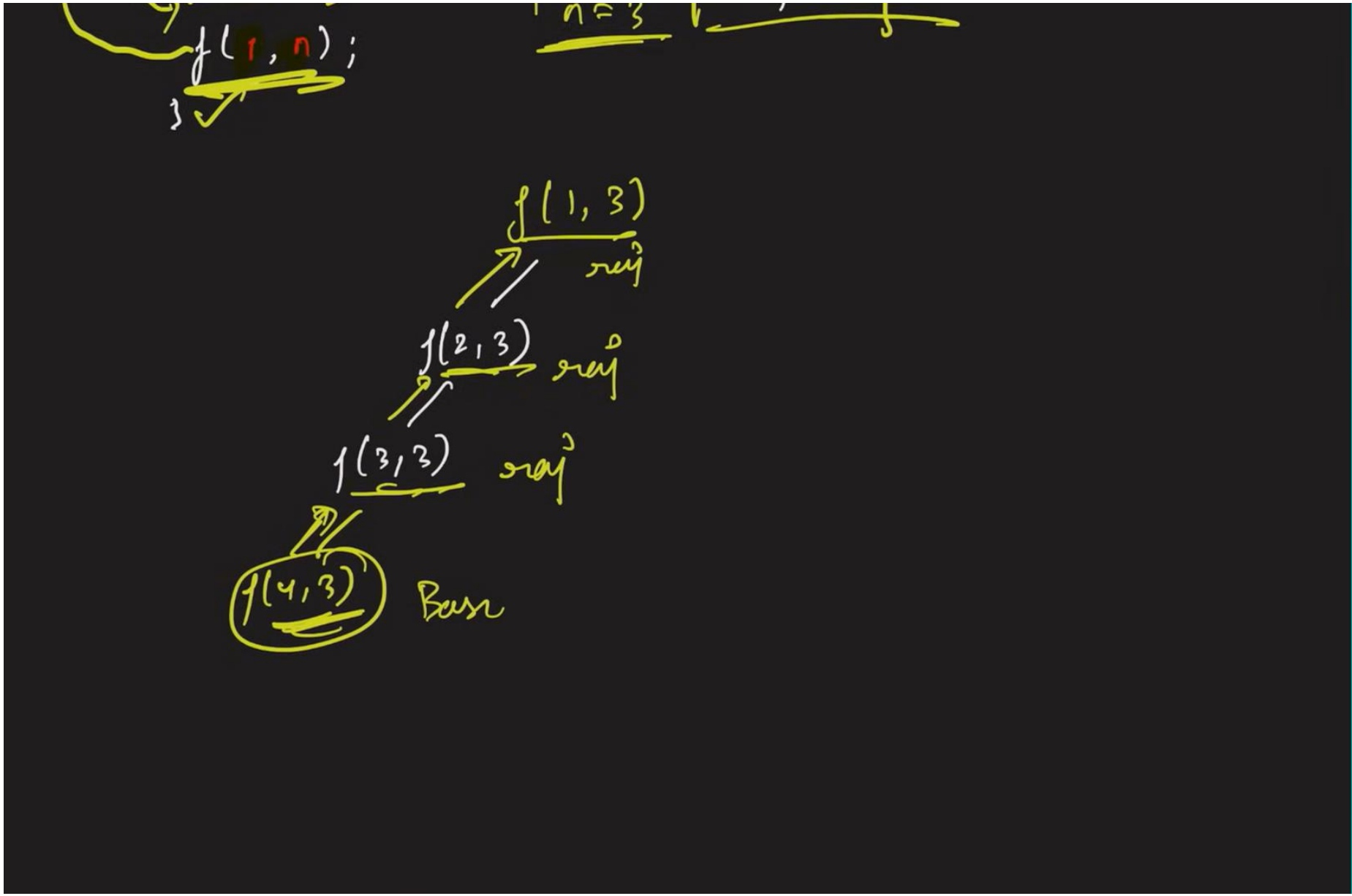


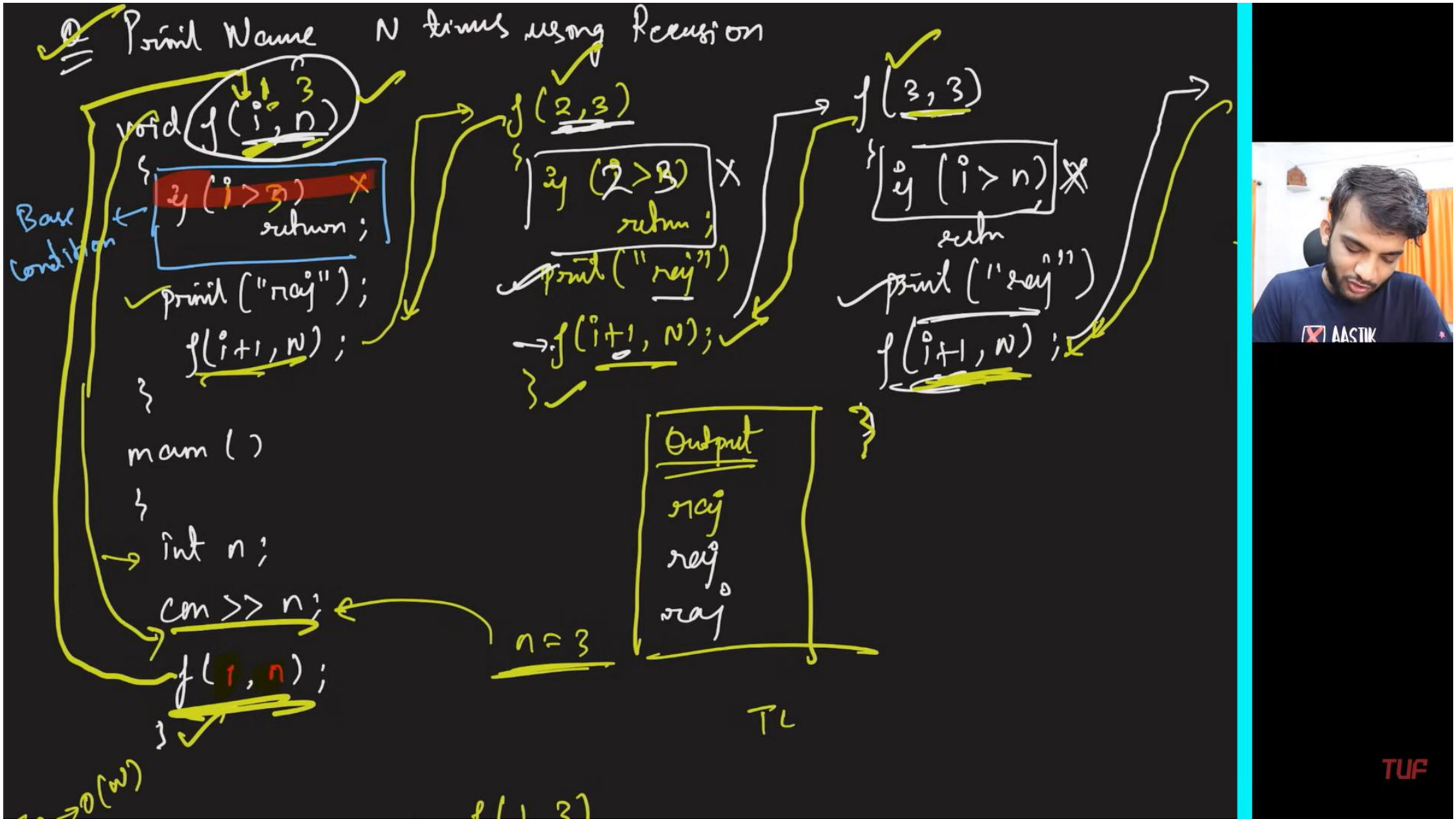
TUF

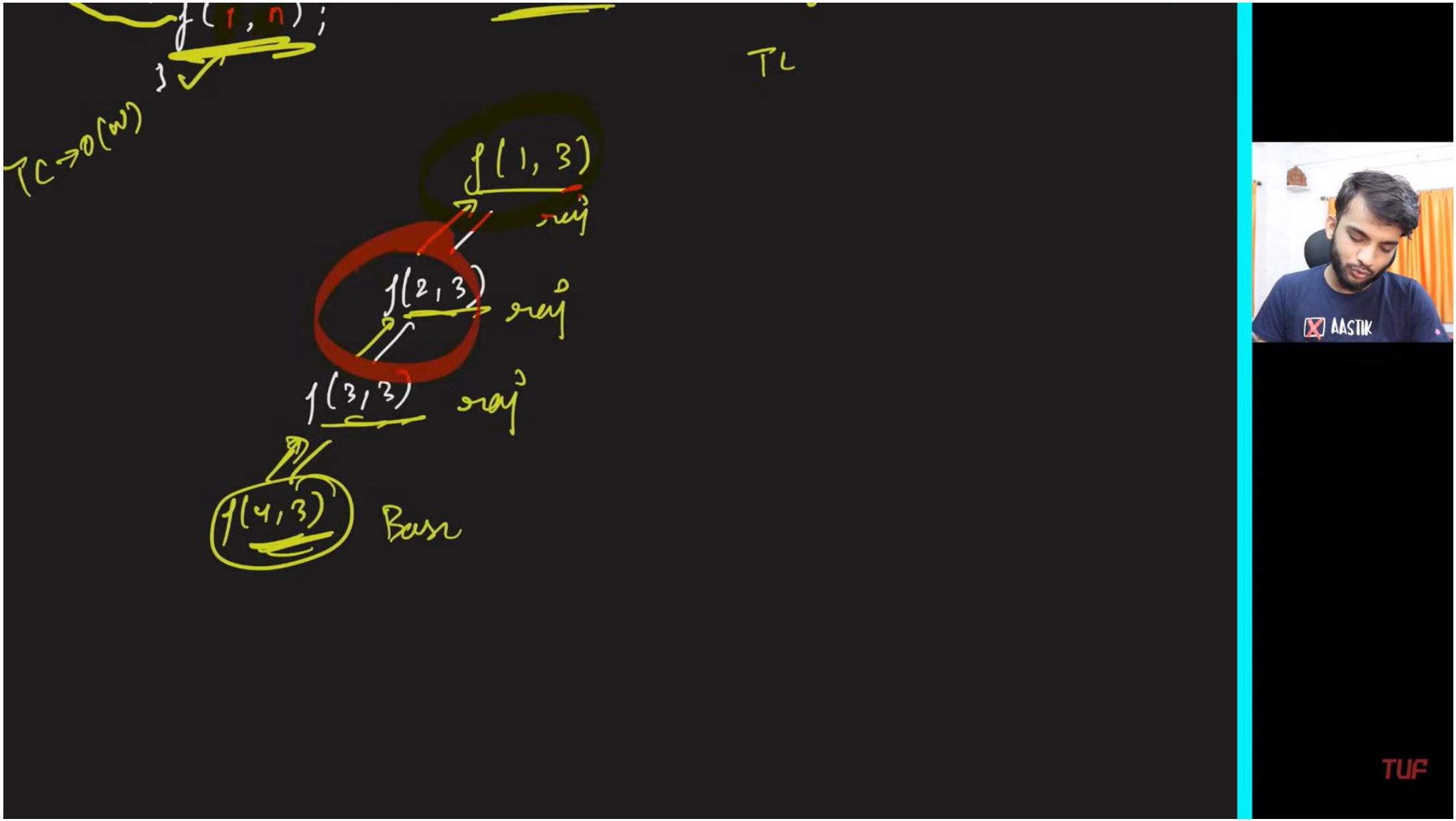


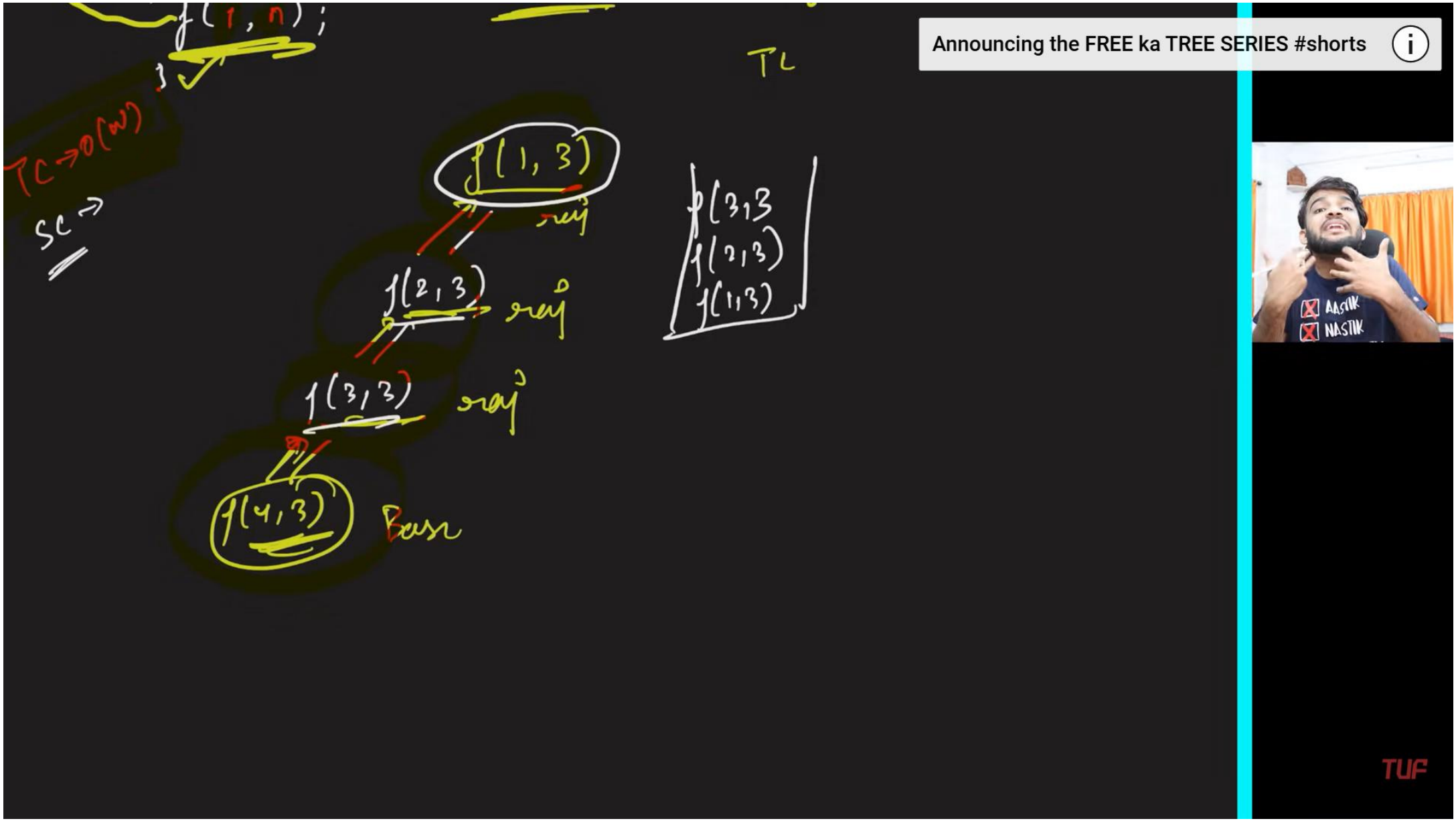


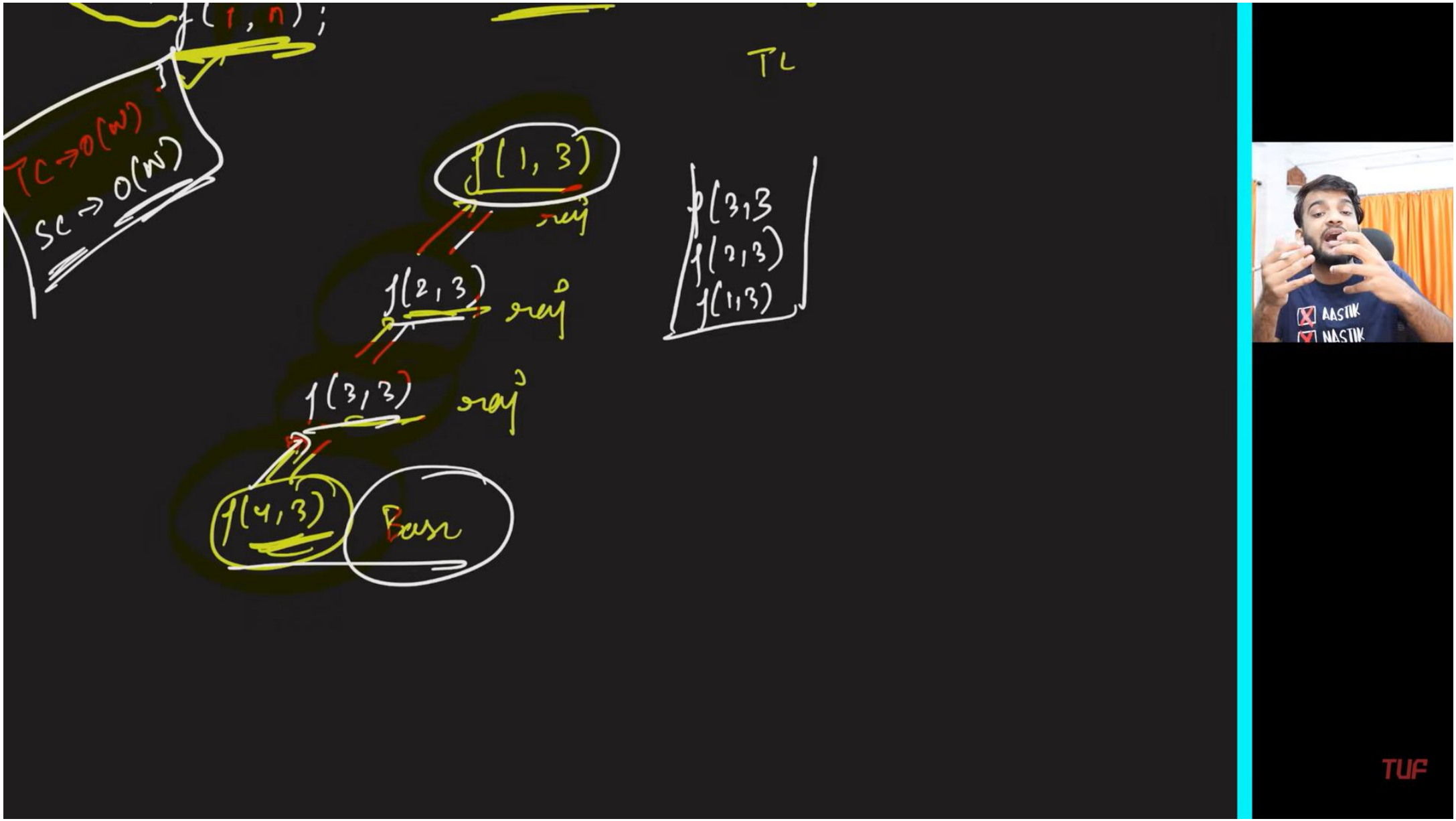
TUF

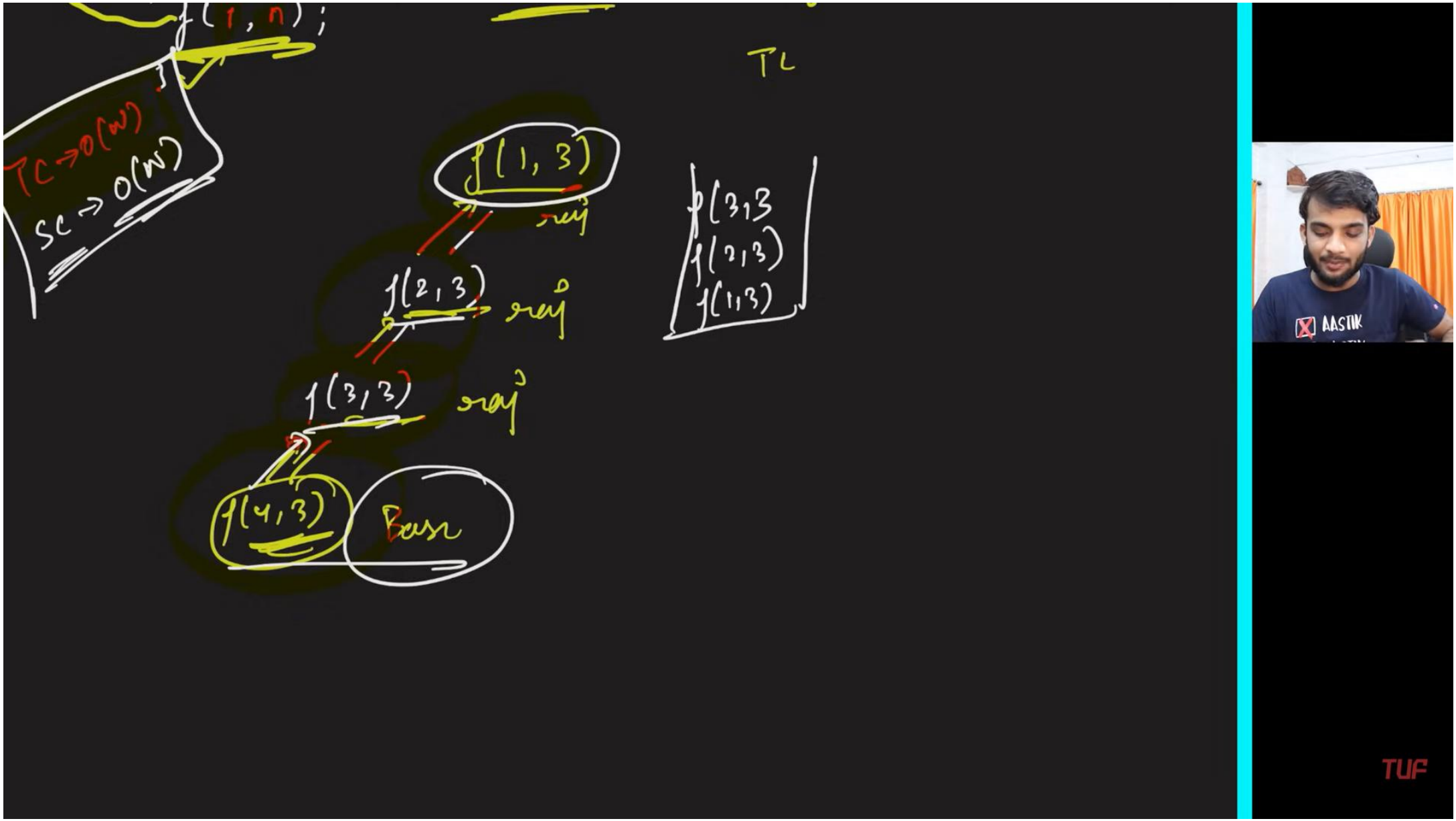

f(1, n);
n=3
f(1, 3)
req
f(2, 3)
req
f(3, 3)
req
f(4, 3)
Base

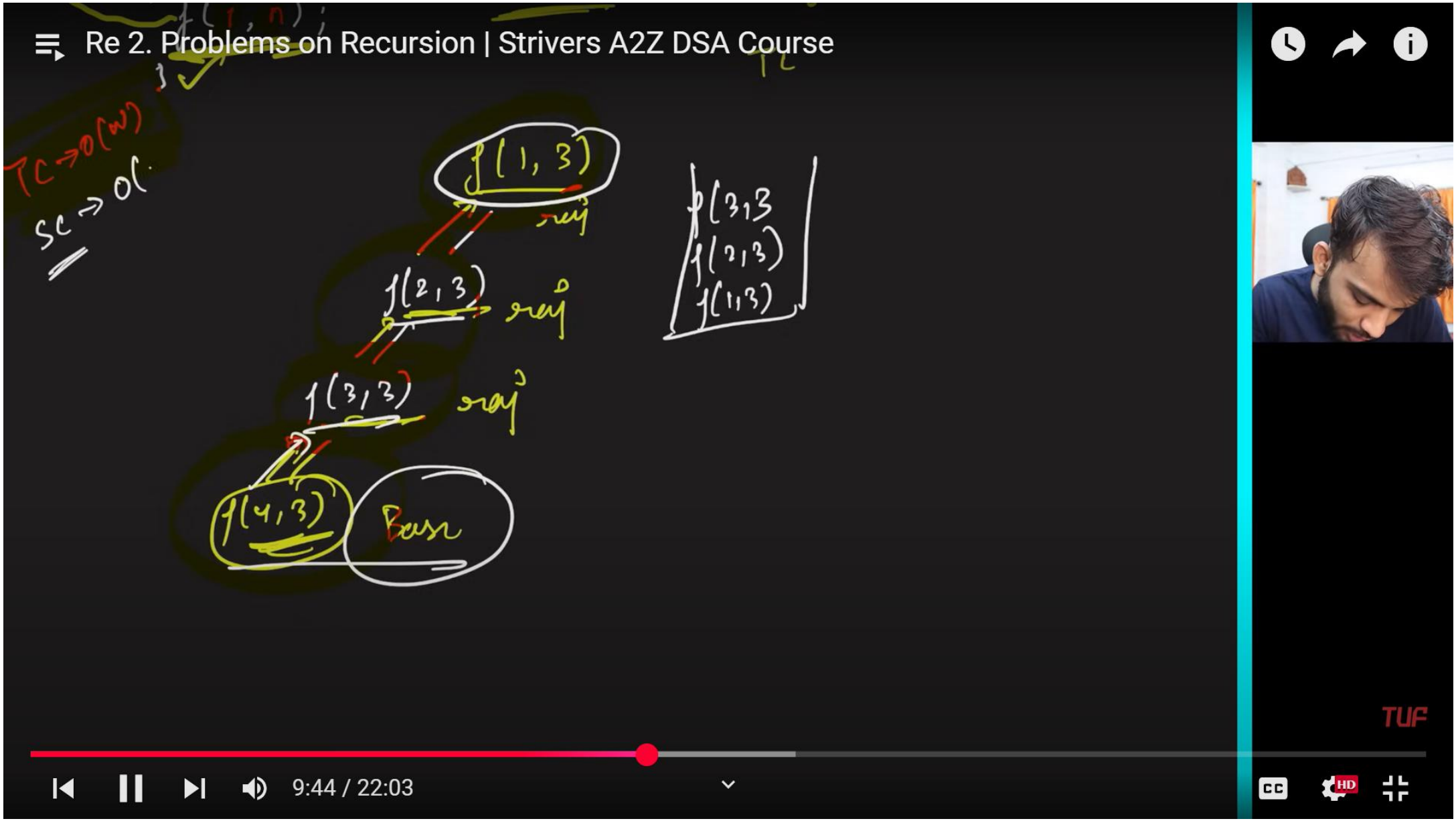

Q Print Name N times using Recursion
void f(i, n)
{
if (i>n)
return;
print("raj");
f(i+1, N);
}
main()
{
int n;
cin >> n;
f(1, n);
}
Base condition
f(1,3)
f(2,3)
f(3,3)
f(4,3)
return
n=3
Output
raj
raj
raj
TL
O(N)
AASTIK
TUF

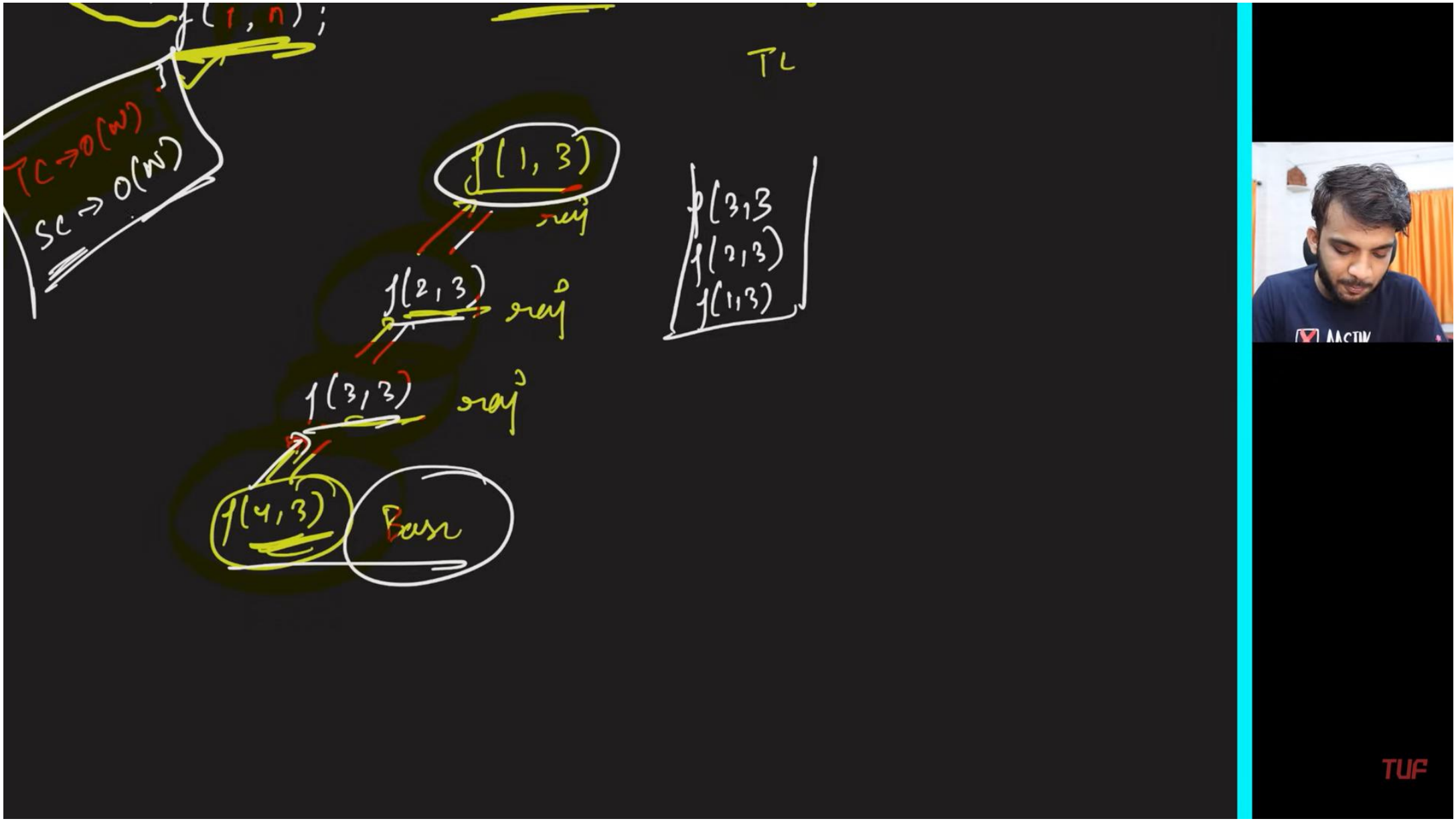

f(1, n);
TC → O(N)
TL
f(1, 3)
recursion
f(2, 3)
recursion
f(3, 3)
recursion
f(4, 3)
Base
AASTIK
TUF

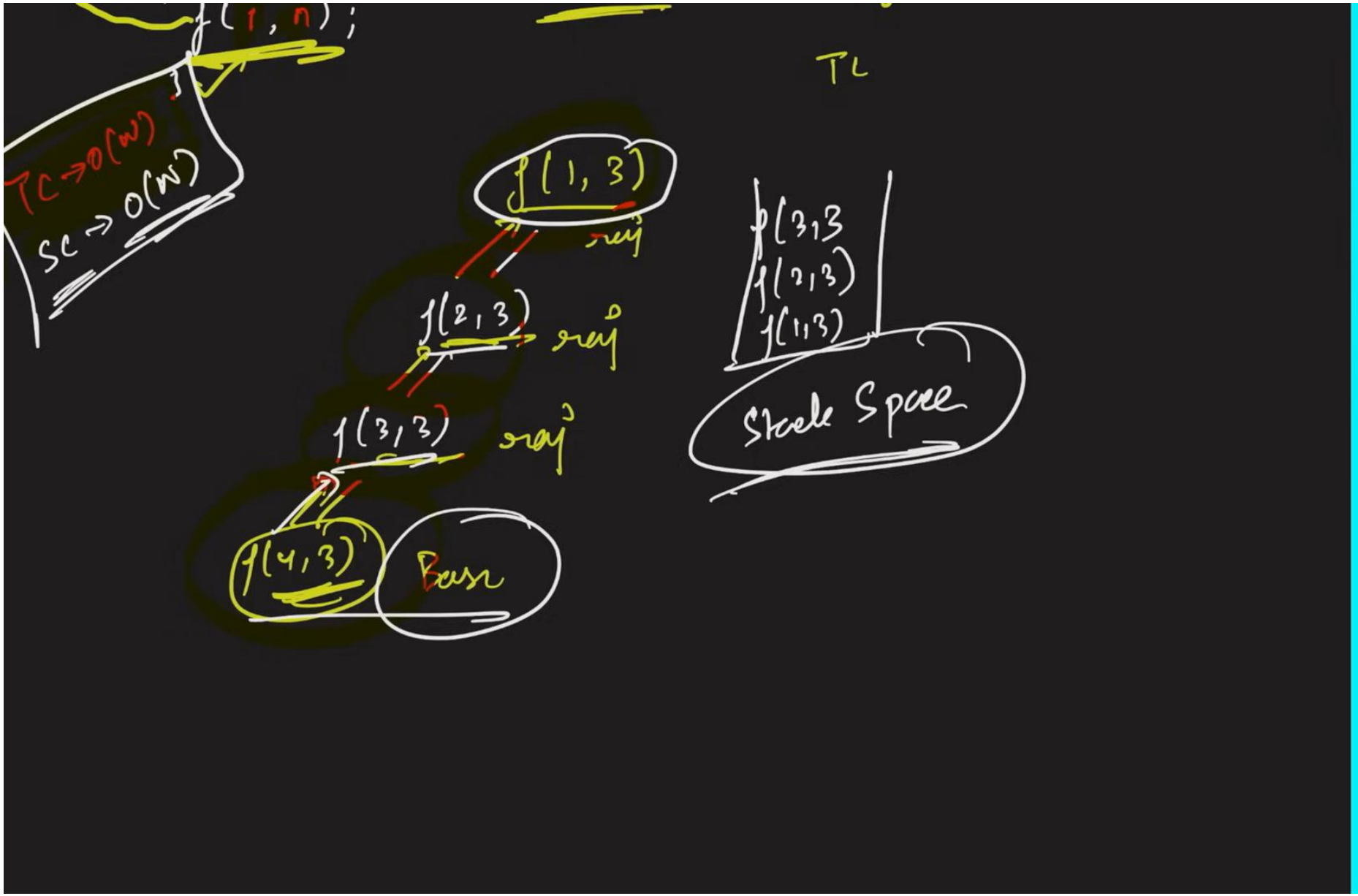

f(1, n);
TC → O(N)
SC →
TL
f(1,3)
req
f(2,3)
req
f(3,3)
req
f(4,3)
Base
f(3,3
f(2,3)
f(1,3)
Announcing the FREE ka TREE SERIES #shorts
AASTIK
NASTIK
TUF


(1, n);
TC → O(N)
SC → O(N)
TL
f(1, 3)
req
f(2, 3)
req
f(3, 3)
req
f(4, 3)
Base
f(3,3
f(2,3)
f(1,3)
AASTIK
NASTIK


(1, n);
TC → O(N)
SC → O(N)
TL
f(1, 3)
req
f(2, 3)
req
f(3, 3)
req
f(4, 3)
Base
f(3,3
f(2,3)
f(1,3)


Re 2. Problems on Recursion | Strivers A2Z DSA Course
TC → O(n)
SC → O(
f(1, 3)
f(2, 3)
f(3, 3)
f(4, 3)
Base
f(3,3
f(2,3)
f(1,3)
TUF
9:44 / 22:03


(1, n);
TC → O(N)
SC → O(N)
TL
f(1, 3)
req
f(2, 3)
req
f(3, 3)
req
f(4, 3)
Base
f(3,3
f(2,3)
f(1,3)


(1, n);
TC → O(N)
SC → O(N)
TL
f(1, 3)
req
f(2, 3)
req
f(3, 3)
req
f(4, 3)
Base
f(3,3
f(2,3)
f(1,3)
Stack Space

(0) Print (1 — N)

N = 4

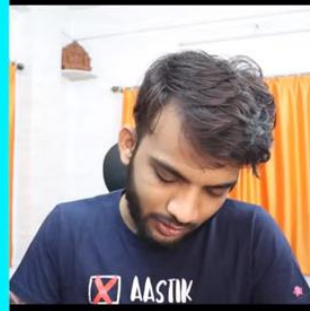
TUF

(6) Print (1 — N) N=4

1	2	3	4

```
f(i, N)
{
  if (i > N)
      return;
  print(i)
  f(i+1, N)
}
```

```
f(i, N)
{
  if (i > N)
      return;
  print(i)
  f(i+1, N)
}
main()
{
```

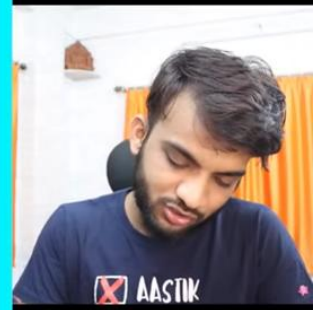


TUF

```
f(i, N)
{
  if (i > N)
      return;
  print(i)
  f(i+1, N)
}
main()
{
  input(n)
  f(1, N)
}
```

Output

1

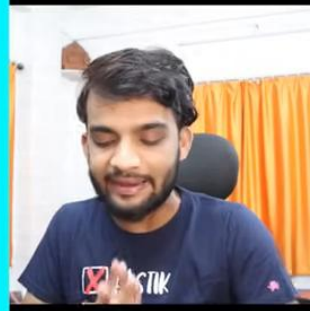


TUF

```
f(i, N)
{
  if (i > N)
      return;
  print(i)
  f(i+1, N)
}
main()
{
  input(n)
  f(1, N)
}
```

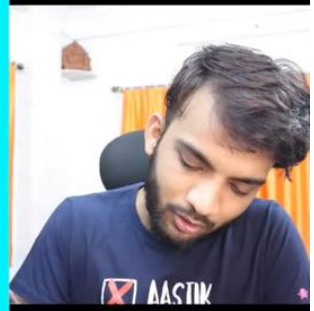
Output

1
2
3
4



(6) Print in terms of $N \rightarrow 1$ $\underline{N=4}$

| 4 3 2 1



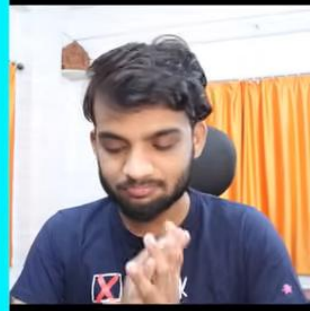
TUF

(6) Print in terms of N → 1 N=4

4	3	2	1

(6) Print in terms of N → 1 N=4

4	3	2	1



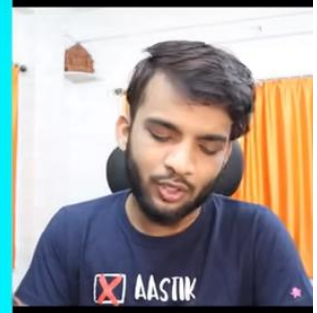
Re 2. Problems on Recursion | Strivers A2Z DSA Course

(0) Print (1 — N) N=4

1	2	3	4

```
f(i, N)
{
  if (i > N)
    return;
  print(i)
  f(i+1, N)
}
main()
```

Output



TUF

11:33 / 22:03

Re 2. Problems on Recursion | Strivers A2Z DSA Course

(8) Print N terms of N -> 1 N=4

4	3	2	1

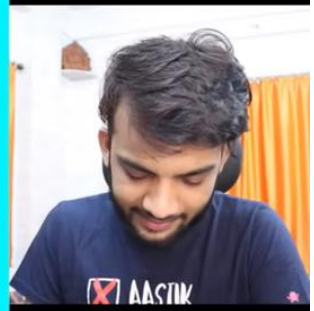
TUF

12:22 / 22:03

```
f(i, N)
{
  if (i < 1)
      return;
  print(i)
  f(i-1, N);
}
```

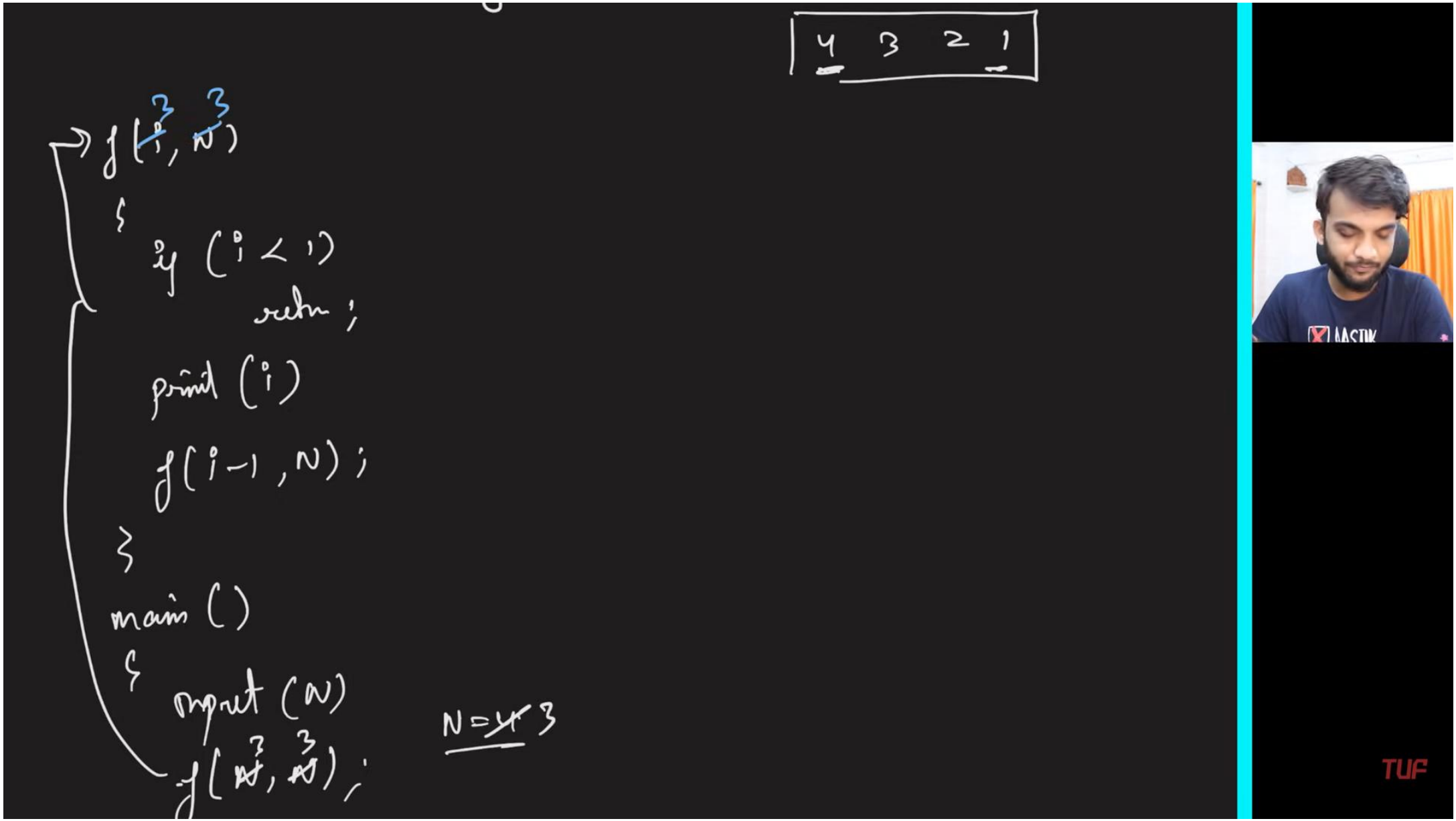
TUF

```
f(i, N)
{
  if (i < 1)
      return;
  print (i)
  f(i-1, N);
}
main ()
{
  input (N)
  .
```



TUF

```
f(i, N)
{
  if (i < 1)
      return;
  print (i)
  f(i-1, N);
}
main ()
{
  input (N)
  f(N, N);
}
```


4 3 2 1
f(3, 3)
{
if (i < 1)
return;
print (i)
f(i-1, N);
}
main ()
{
input (N)
f(3, 3);
N=3

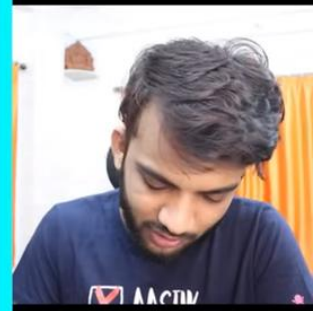
4	3	2	1

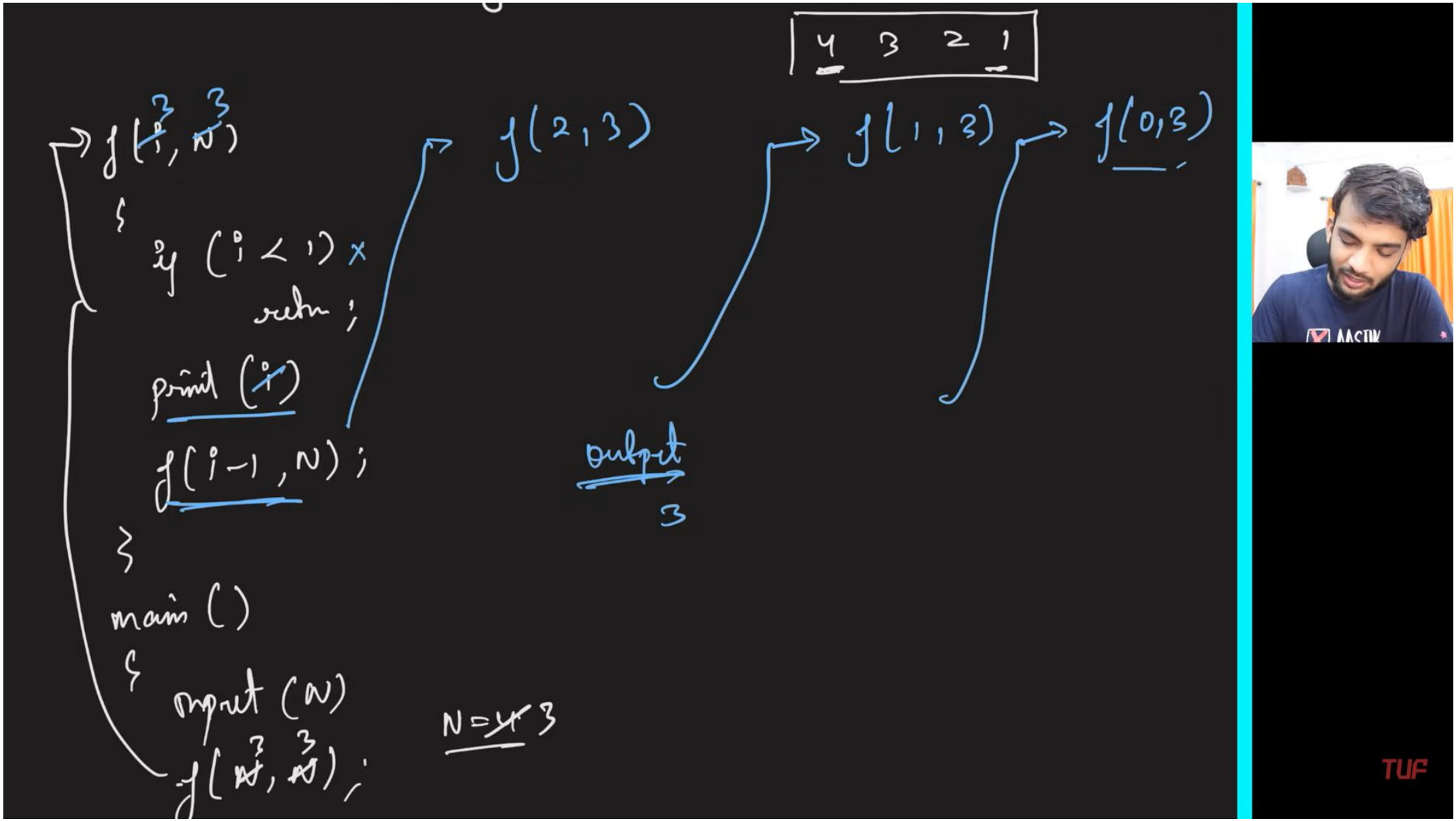
```
f(3, 3)
{
    if (i < 1)
        return;
    print(i)
    f(i-1, N);
}
main()
{
    input(N)
    f(3, 3);
}
```

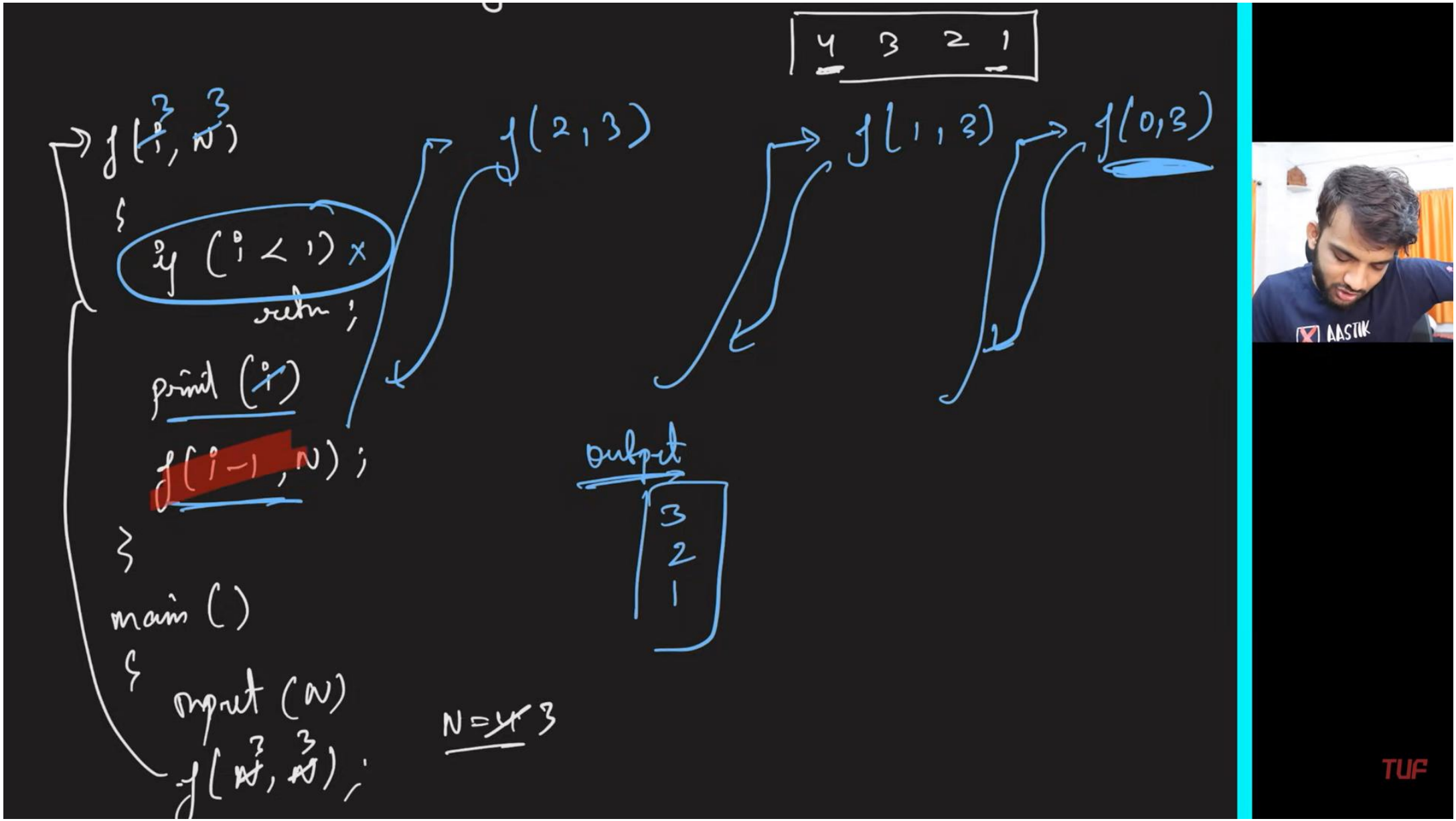
N = 4 → 3

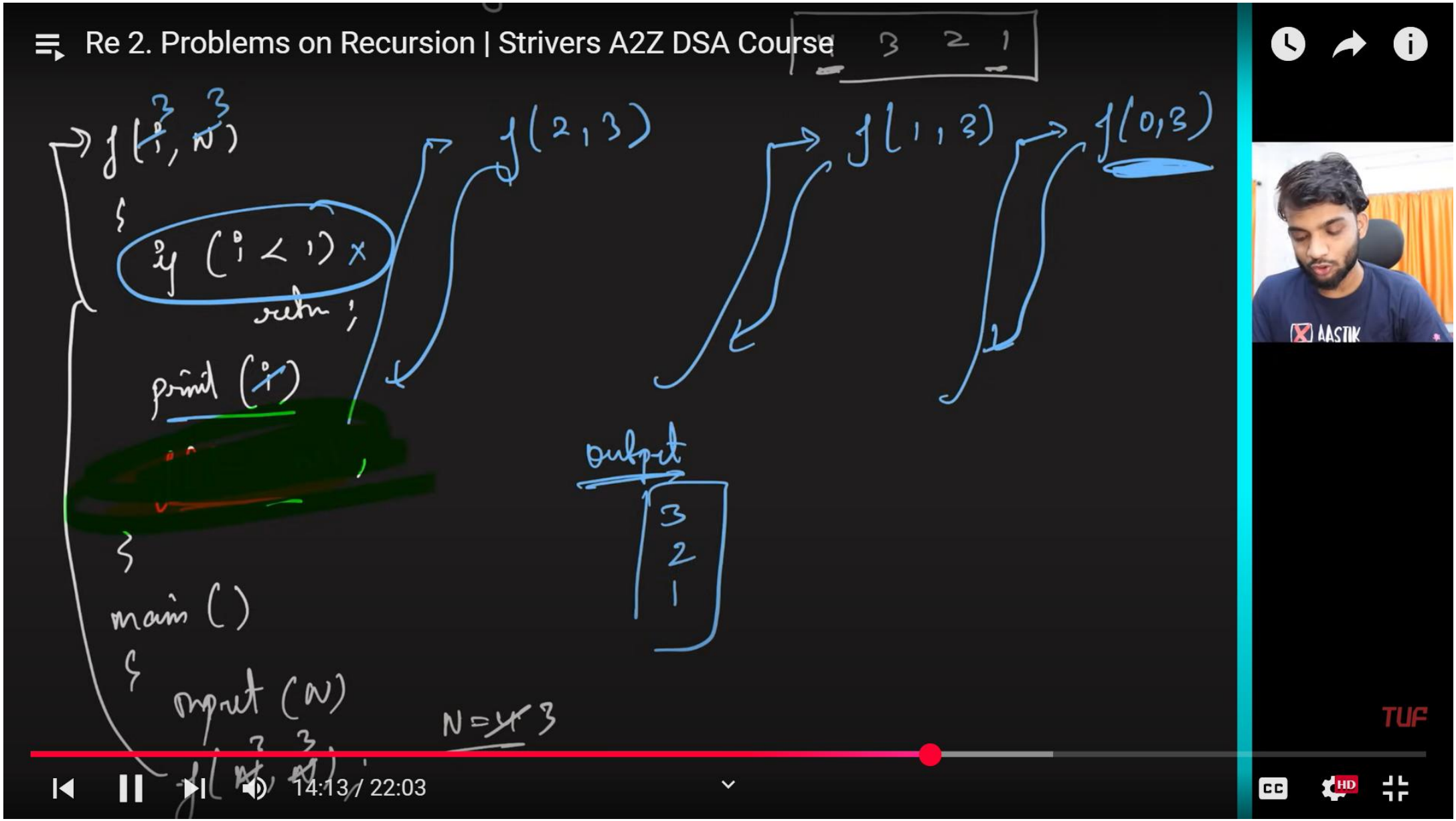
output

3




4 3 2 1
f(3, 3)
{
if (i < 1)
return;
print (i)
f(i-1, N);
}
main ()
{
input (N)
f(3, 3);
f(2,3)
f(1,3)
f(0,3)
output
3
N=3


4 3 2 1
f(3, 3)
f(2,3)
f(1,3)
f(0,3)
if (i < 1) x
return ;
print (i)
f(i-1, N) ;
output
3
2
1
main ()
input (N)
f(3, 3) ;
N = 3


Re 2. Problems on Recursion | Strivers A2Z DSA Course
3 2 1
f(3, N)
f(2,3)
f(1,3)
f(0,3)
if (i < 1) x
return ;
print (i)
output
3
2
1
main ()
input (N)
N = 3
14:13 / 22:03
AASTIK
TUF

asic Recursion Problems

- Print Name N times
- Print Linearly from 1 to N
- Print from N to 1.
- Print Linearly from 1 to N (But By Backtracking)
- Print from N to 1 (By Backtracking)

Print Name N times using Recursion

(or) Print from 1 to N.

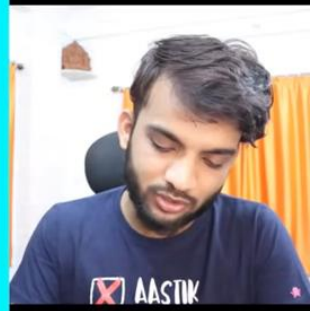
f(i+1, N) X

TUF

```
f(i, N)
{
  if (i < 1)
    return;

  f(i-1, N);
  print(i);
}

main()
{
```

TUF

```
f(i, N)
{
  if (i < 1)
    return;

  f(i-1, N);
  print(i);
}

main()
{
  input(n)
  f(N, N);
}
```



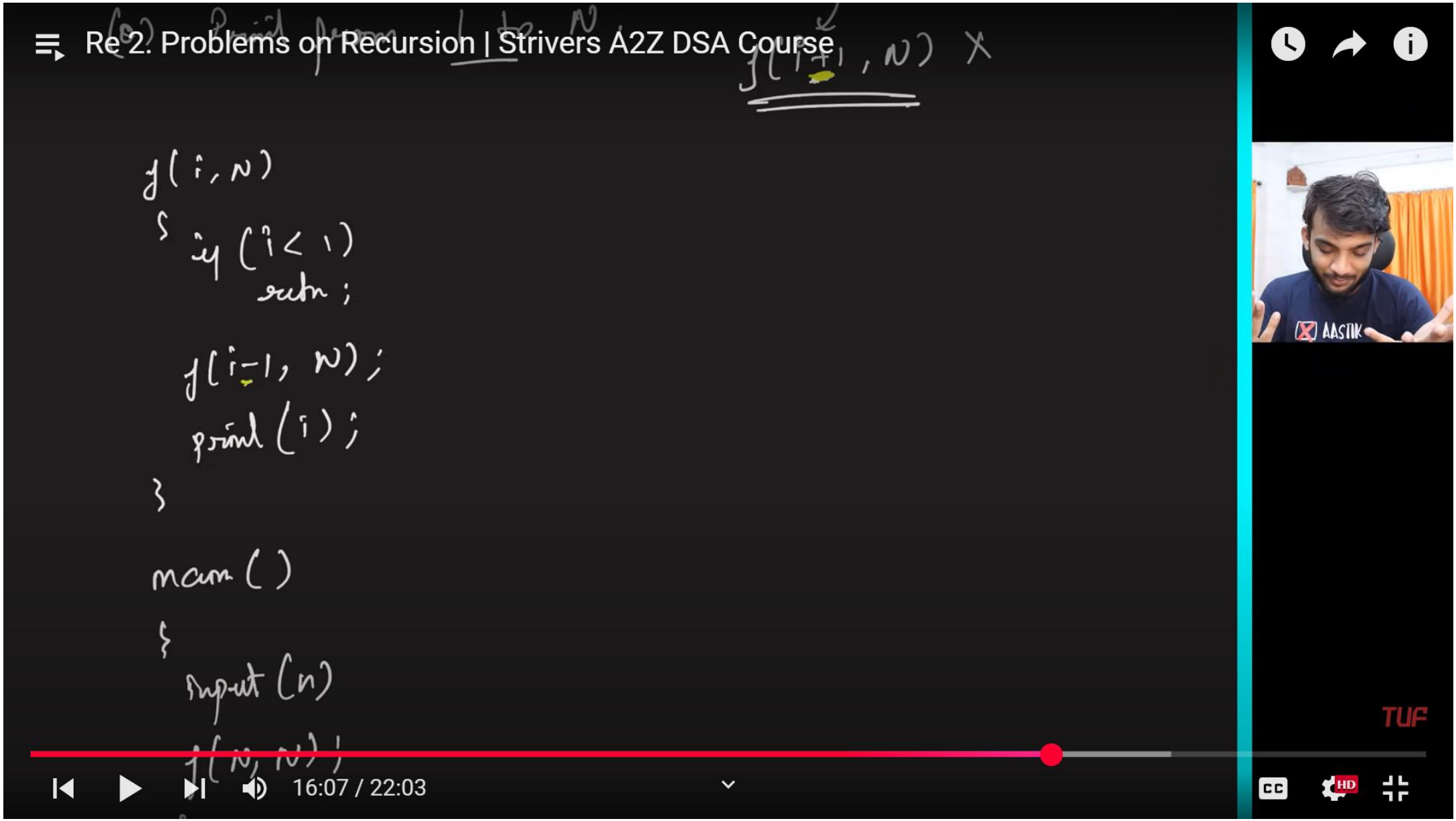
TUF

(8) Print from 1 to N.

f(i+1, N) ✗

```
f(i, N)
{
  if (i < 1)
    return;
  f(i-1, N);
  print(i);
}

main()
{
  input(n)
  f(N, N);
```

Re 2. Problems on Recursion | Strivers A2Z DSA Course

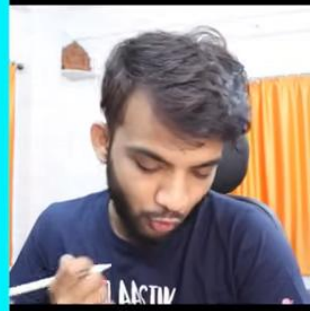


(8) Print from 1 to N.

f(i+1, N) X

```
f(3, 3)
{
  if (i < 1)
    return;

  f(i-1, N);
  print(i);
}

main()
{
  input(n)   <- 3
  f(3, 3);
}
```

Output



TUF

(8) Print from 1 to N.

```
f(i, N)
{
  if (i < 1)
    return;
  f(i-1, N);
  print(i);
}

main()
{
  input(n)
  f(n, N);
}
```

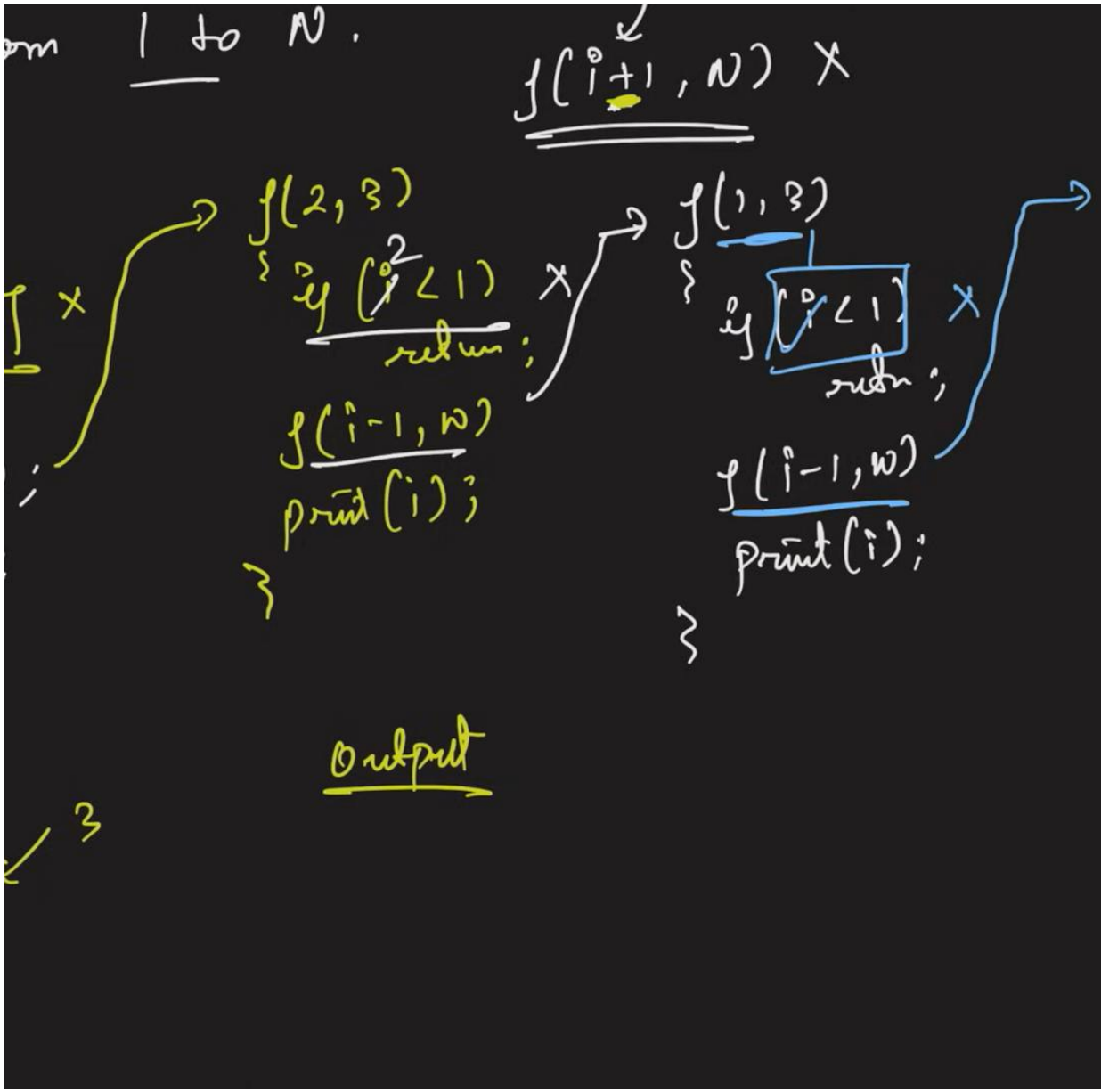
f(i+1, N) ✗

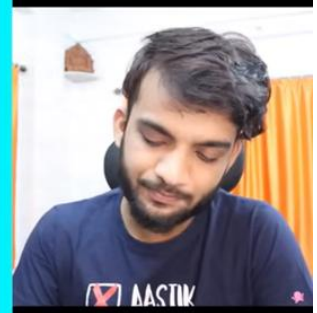
f(2, 3)
{
if (i < 1) ✗
return;
f(i-1, N)
print(i);
}

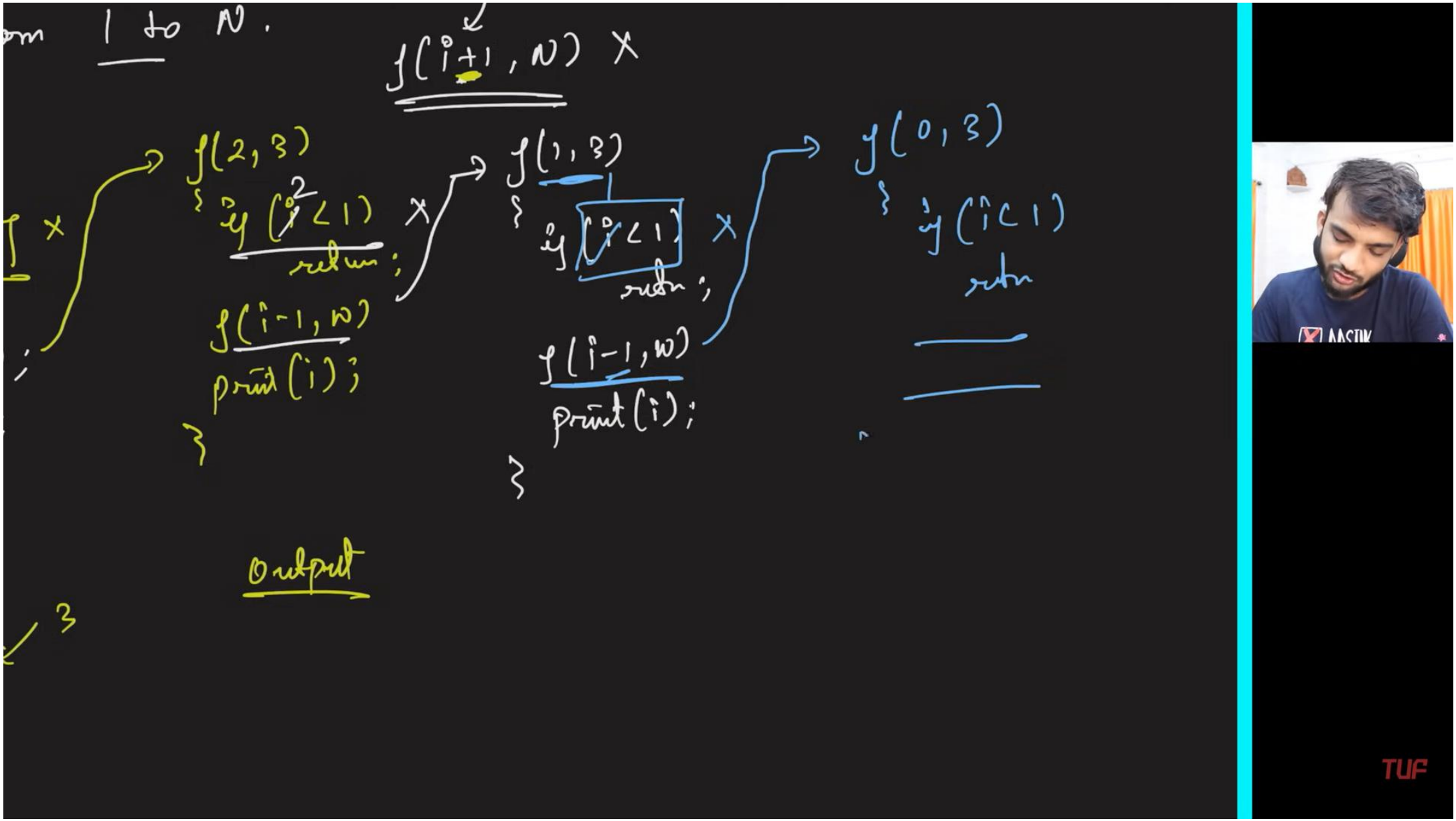
f(

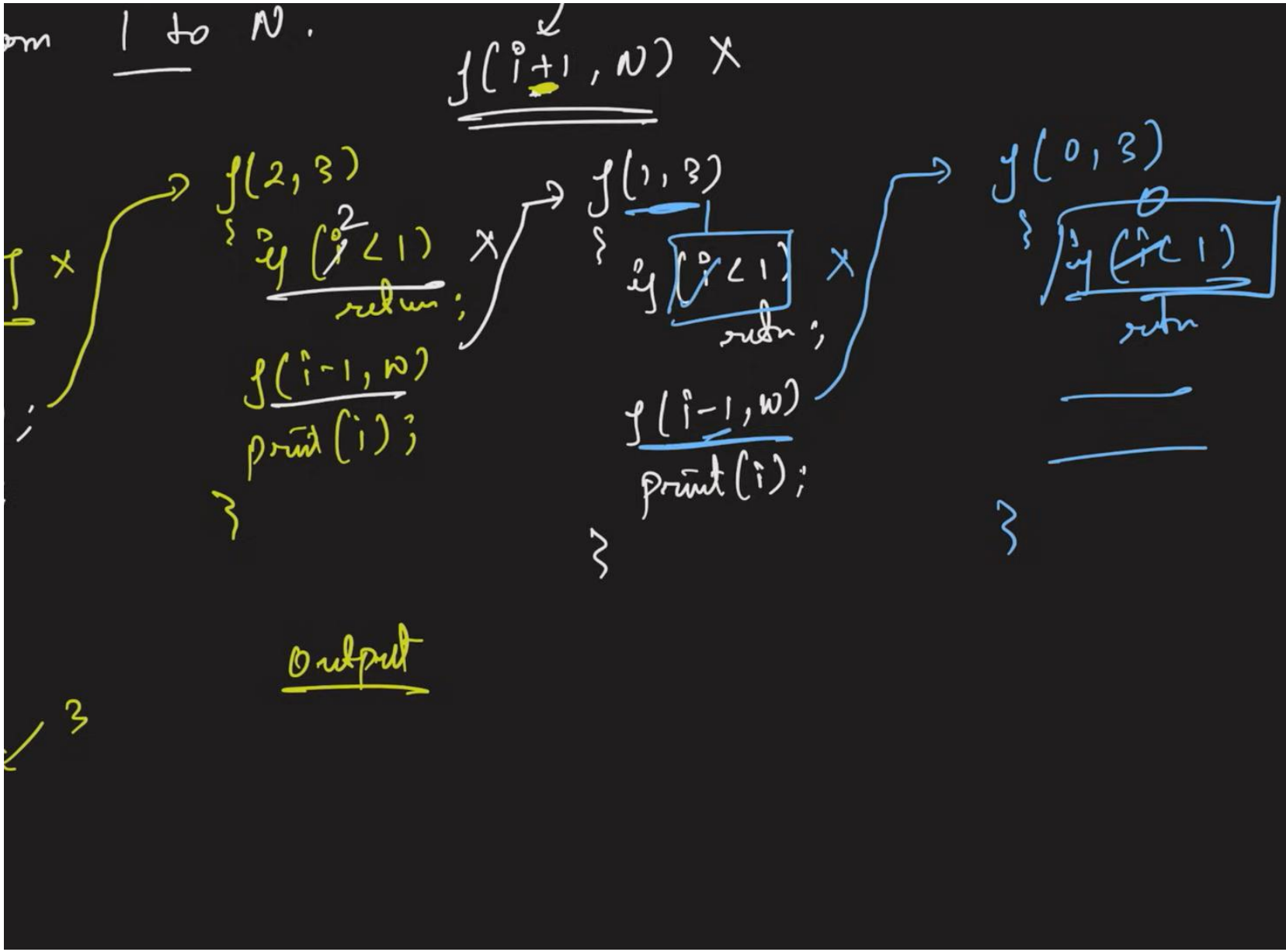
Output

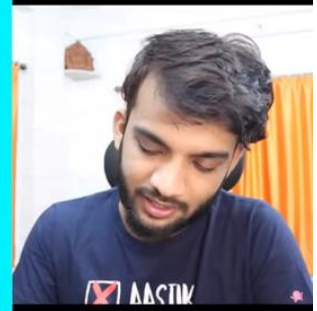
TUF

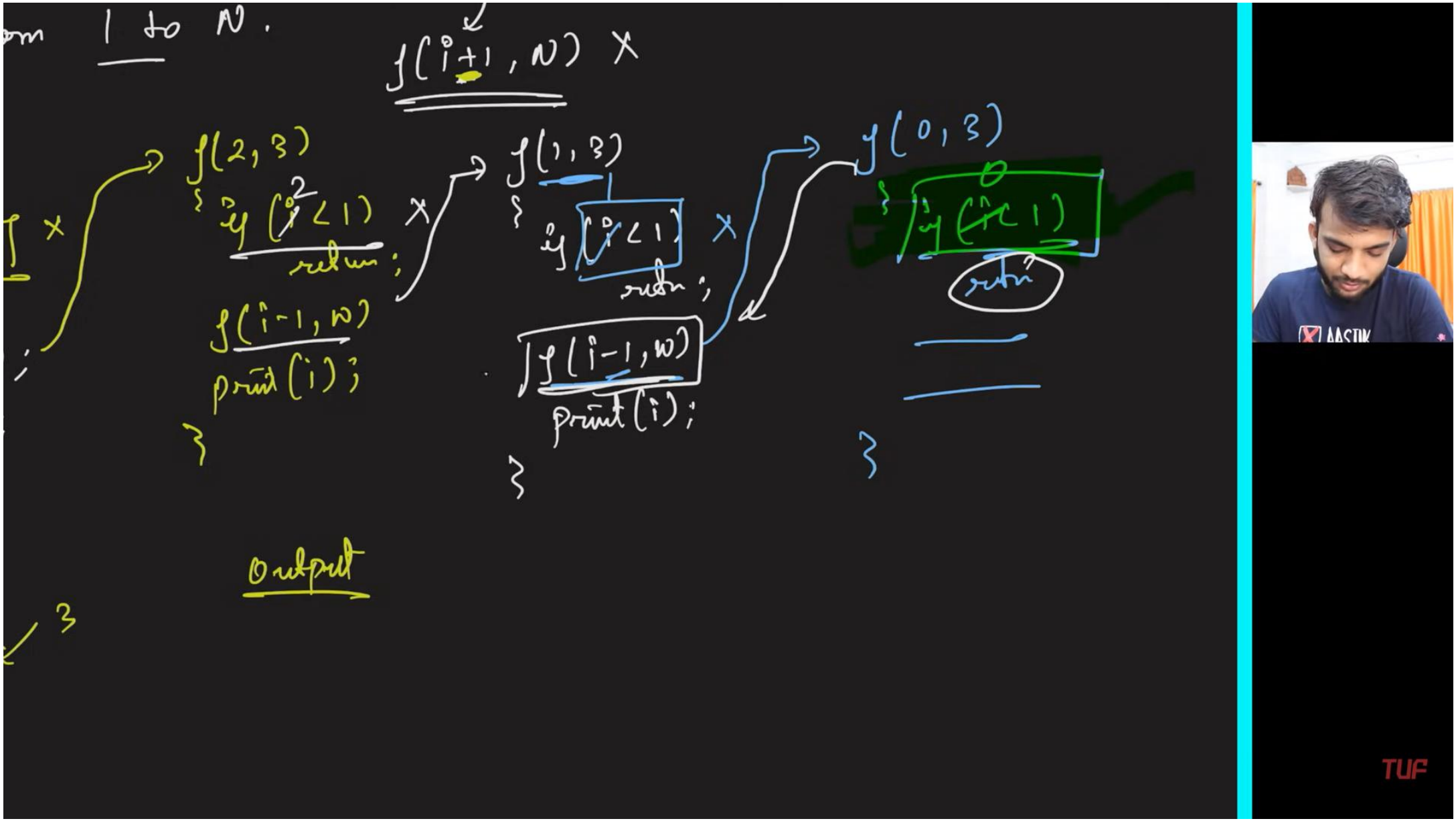

om 1 to N.
f(i+1, N) X
f(2, 3)
if (i^2 < 1)
return;
f(i-1, N)
print(i);
f(1, 3)
if (i^2 < 1)
return;
f(i-1, N)
print(i);
Output
3

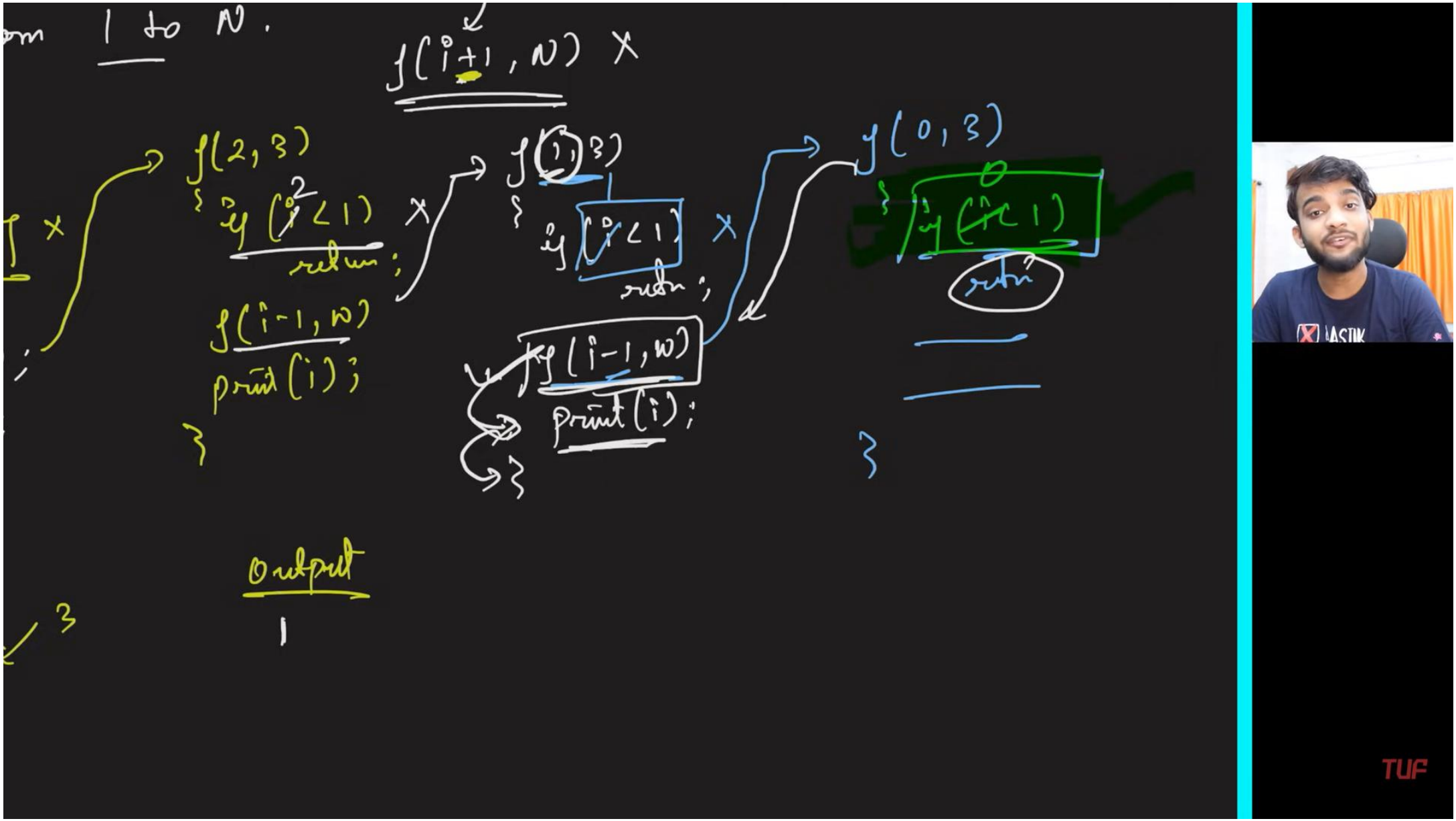

AASTIK

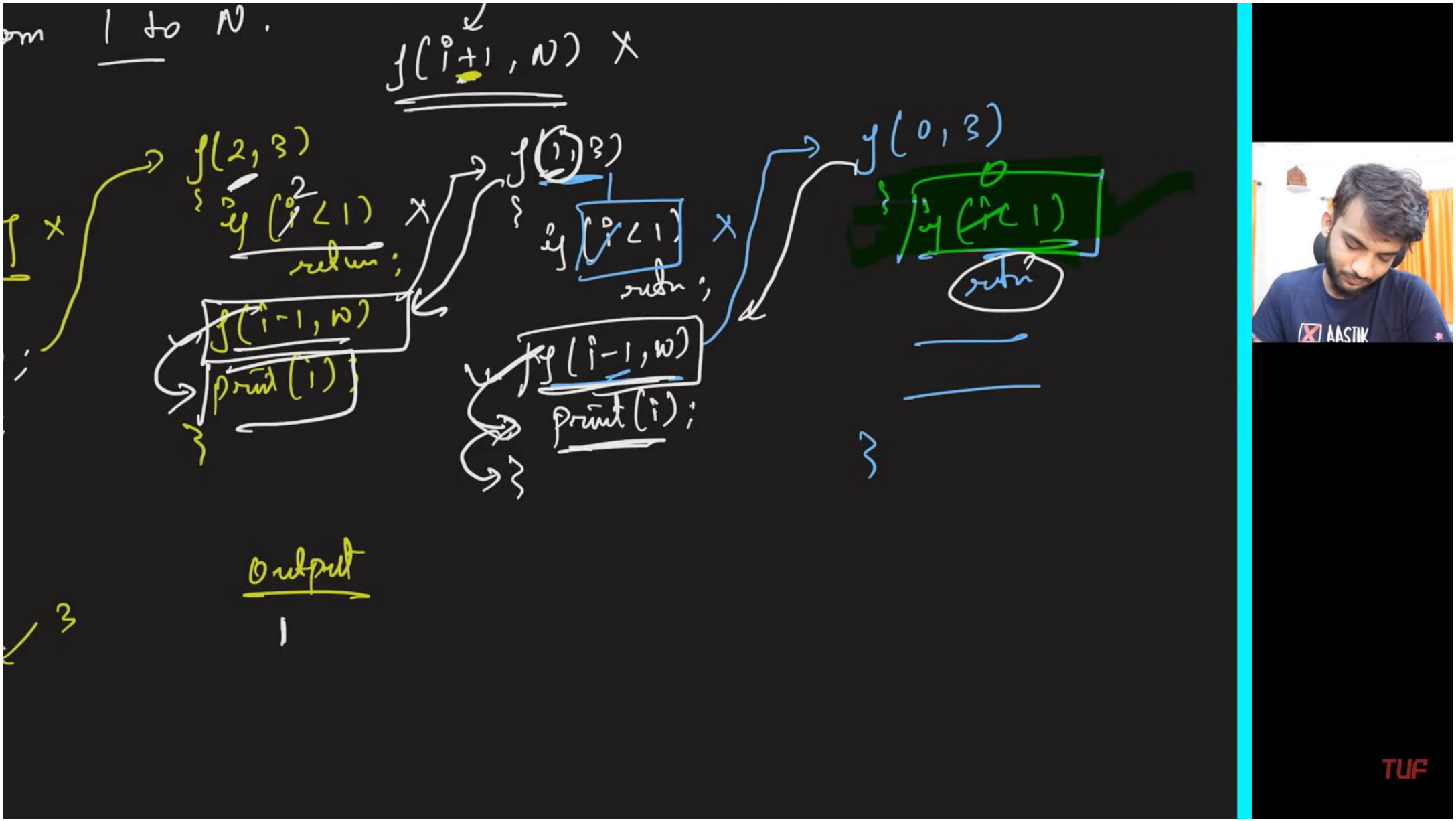

om 1 to N.
f(i+1, N) X
f(2, 3)
if (i^2 < 1)
return;
f(i-1, N)
print(i);
f(1, 3)
if (i < 1)
return;
f(i-1, N)
print(i);
f(0, 3)
if (i < 1)
return
Output
3


om 1 to N.
f(i+1, N) X
f(2, 3)
if (i^2 < 1)
return;
f(i-1, N)
print(i);
f(1, 3)
if (i < 1)
return;
f(i-1, N)
print(i);
f(0, 3)
if (i < 1)
return
Output
3

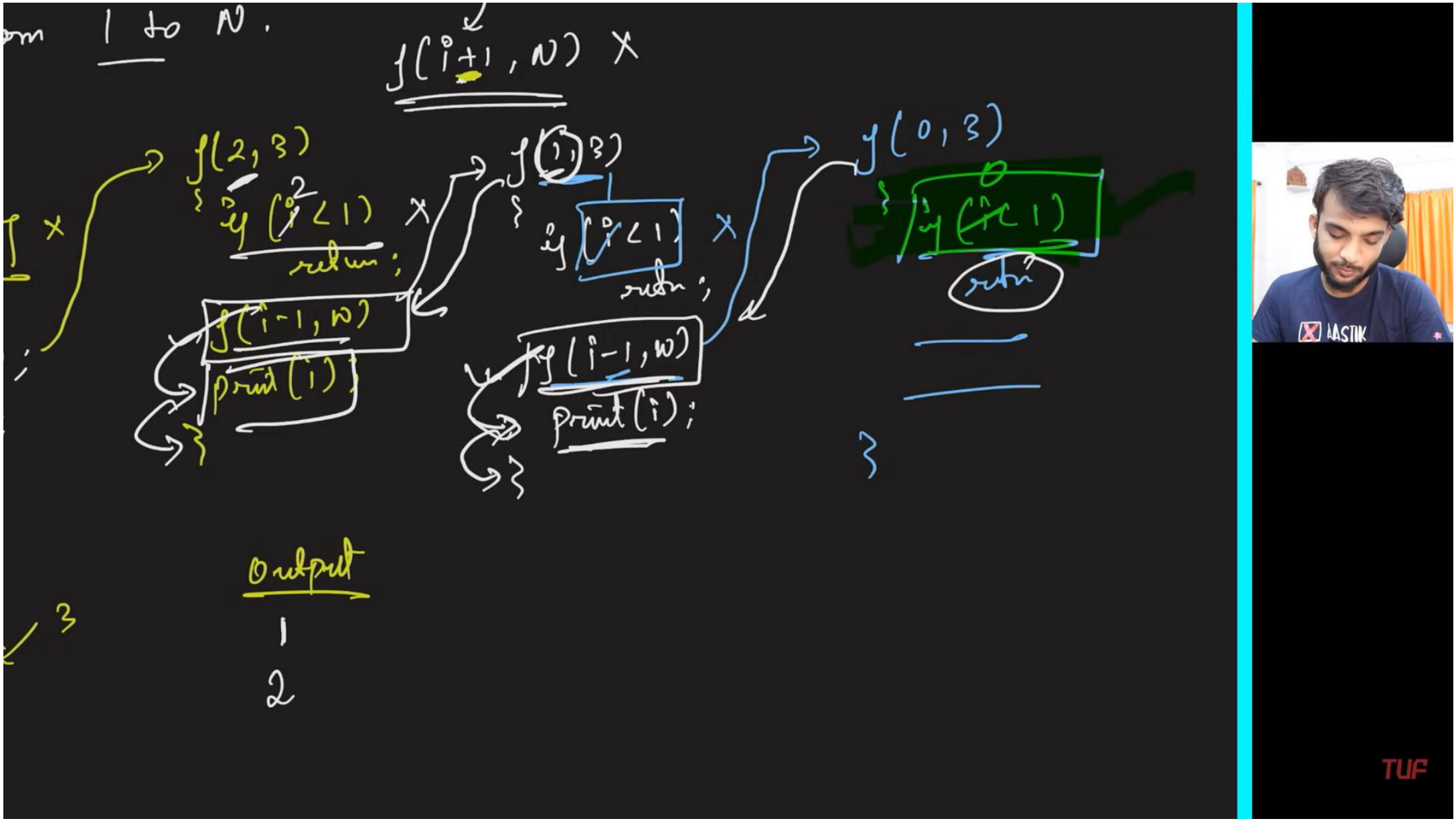

AASTIK


om 1 to N.
f(i+1, N) X
f(2, 3)
if (i^2 < 1)
return;
f(i-1, N)
print(i);
f(1, 3)
if (i < 1)
return;
f(i-1, N)
print(i);
f(0, 3)
if (i < 1)
return
Output
3

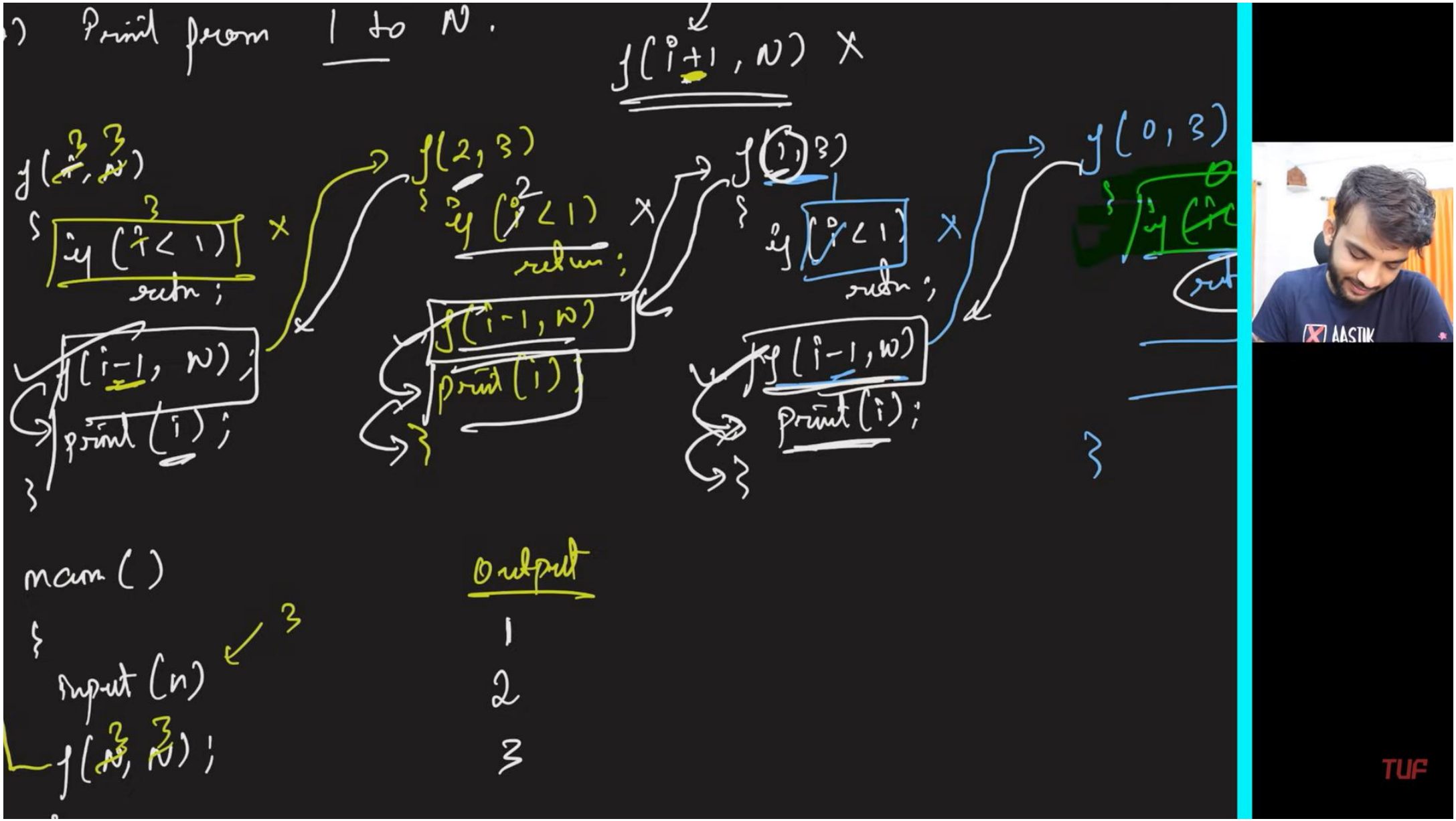

om 1 to N.
f(i+1, N) X
f(2, 3)
if (i^2 < 1)
return;
f(i-1, N)
print(i);
f(1, 3)
if (i < 1)
return;
f(i-1, N)
print(i);
f(0, 3)
if (i < 1)
return
Output
1
3

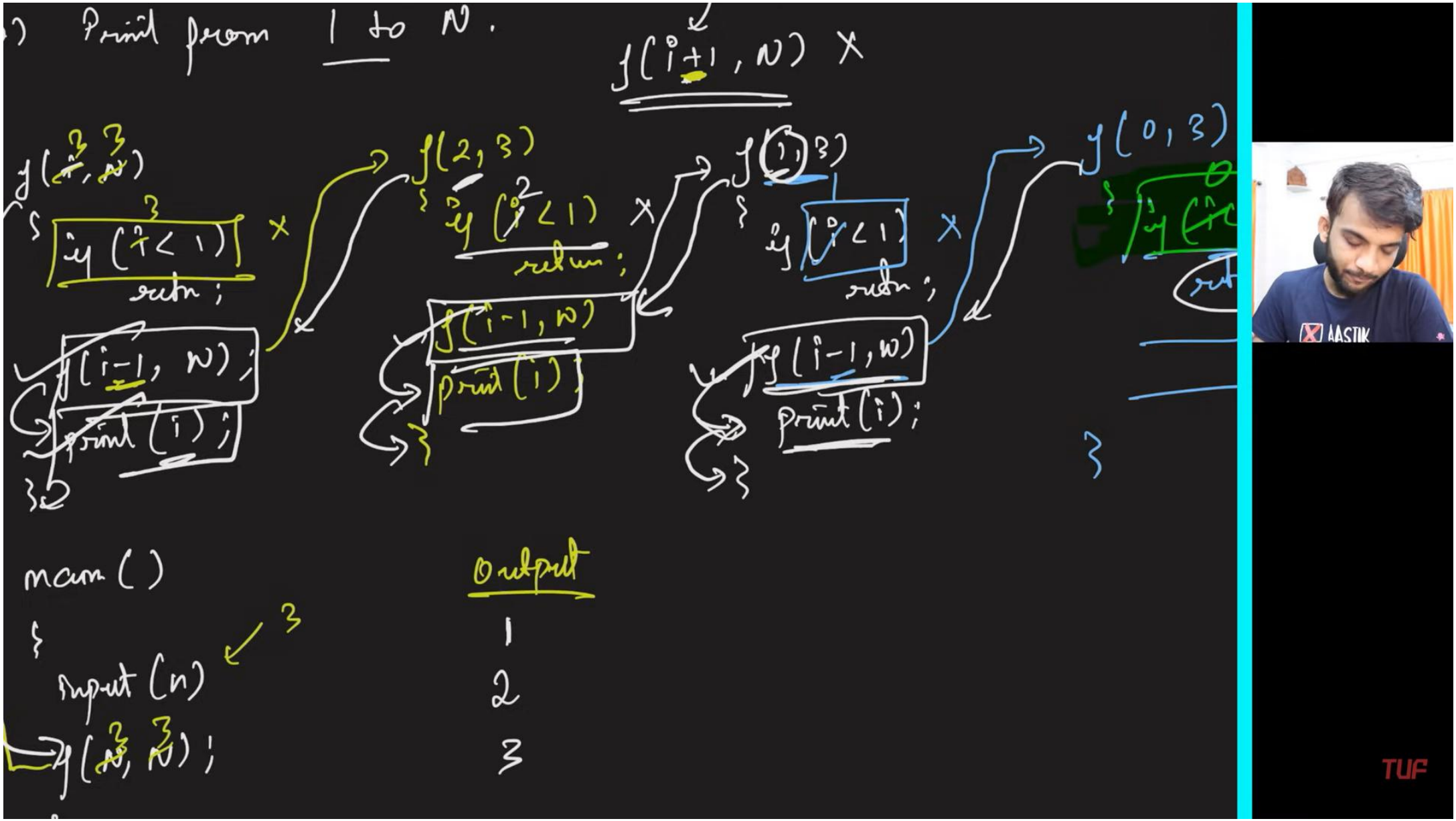

1 to N.
f(i+1, N)
f(2, 3)
if (i<1)
return;
f(i-1, N)
print(i);
f(1, 3)
if (i<1)
return;
f(i-1, N)
print(i);
f(0, 3)
if (i<1)
return
Output
1
3

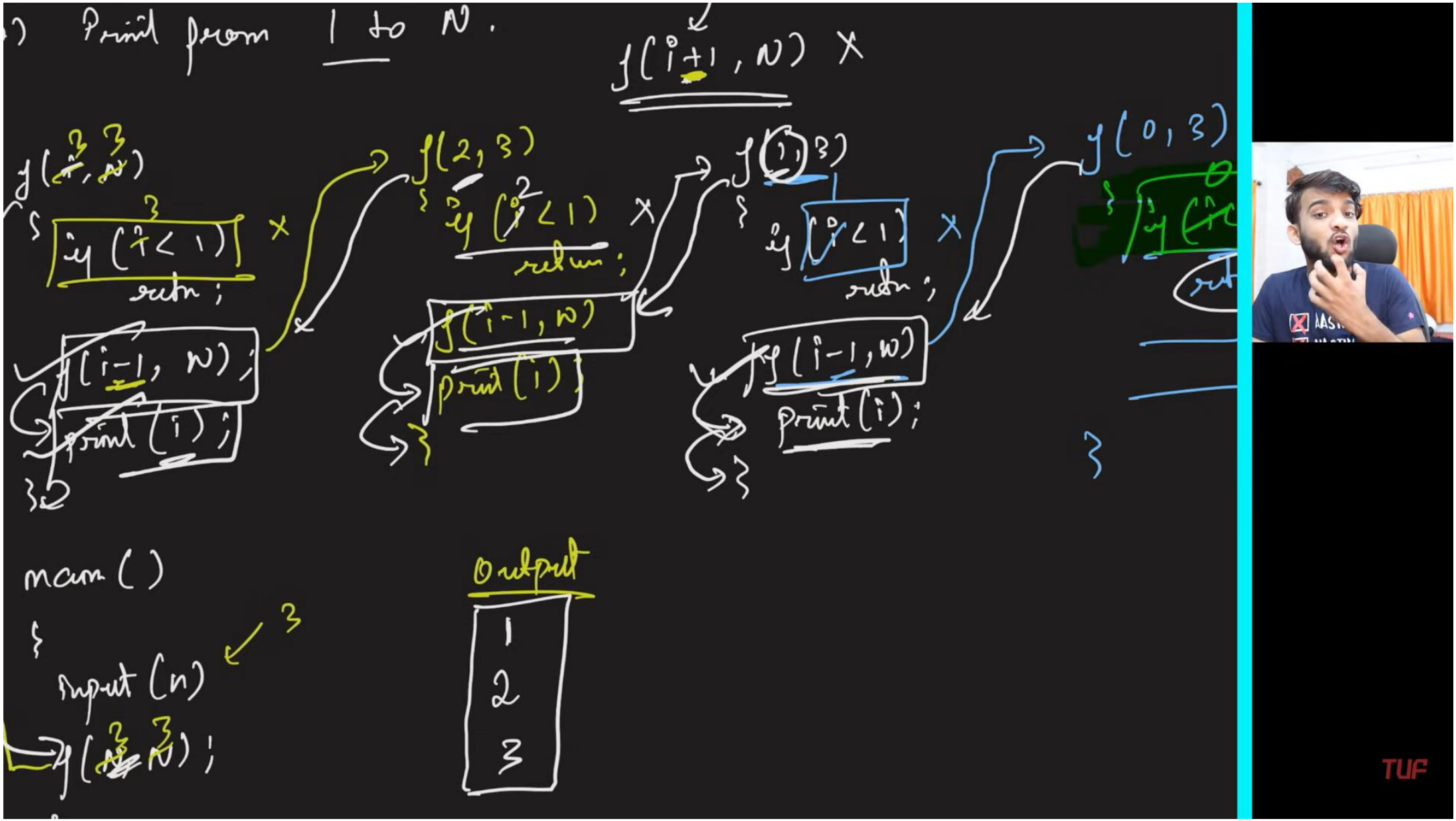
AASTIK

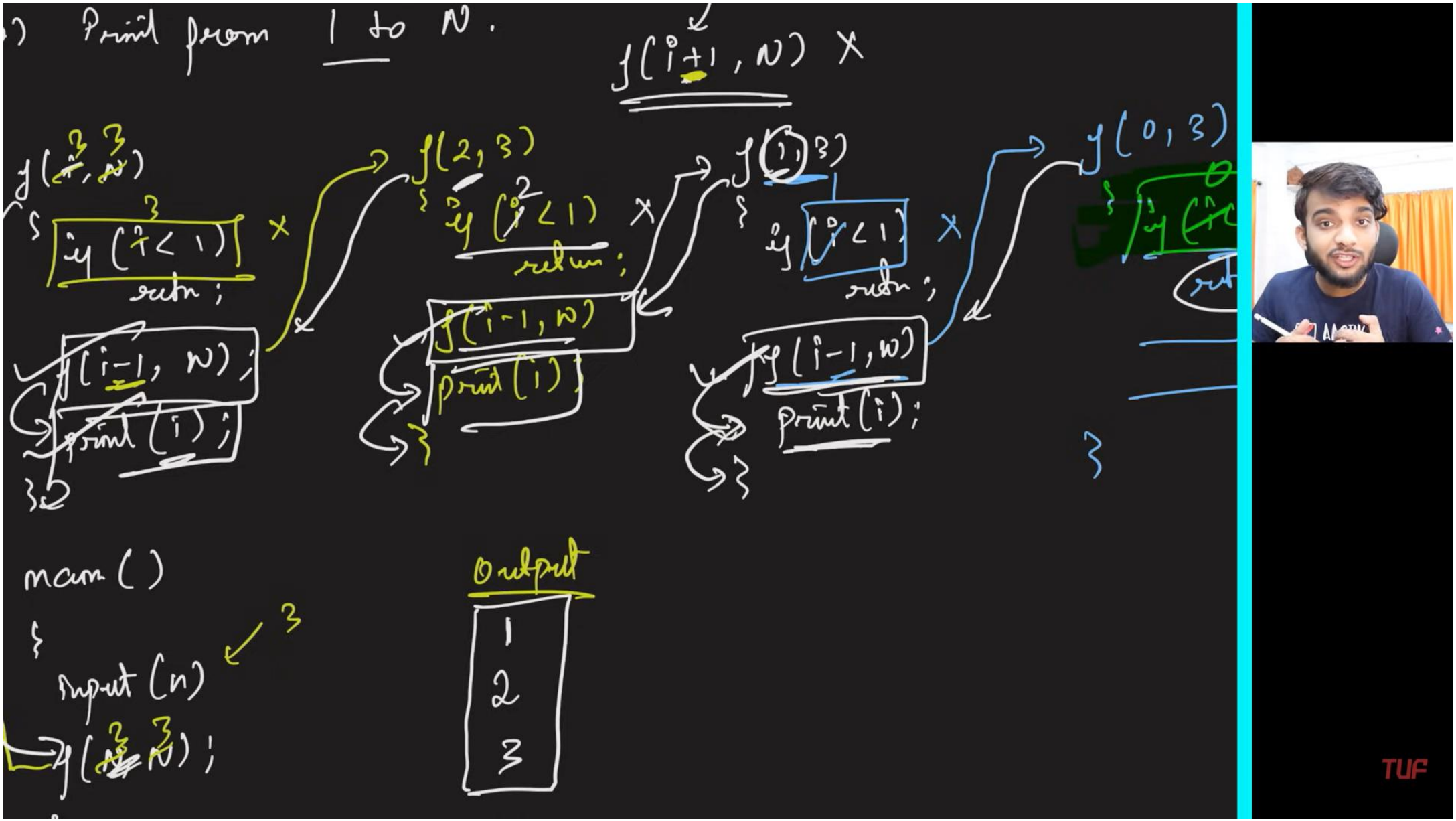

1 to N.
f(i+1, N) X
f(2, 3)
if (i<1)
return;
f(i-1, N)
print(i);
f(1, 3)
if (i<1)
return;
f(i-1, N)
print(i);
f(0, 3)
if (i<1)
return
Output
1
2
3

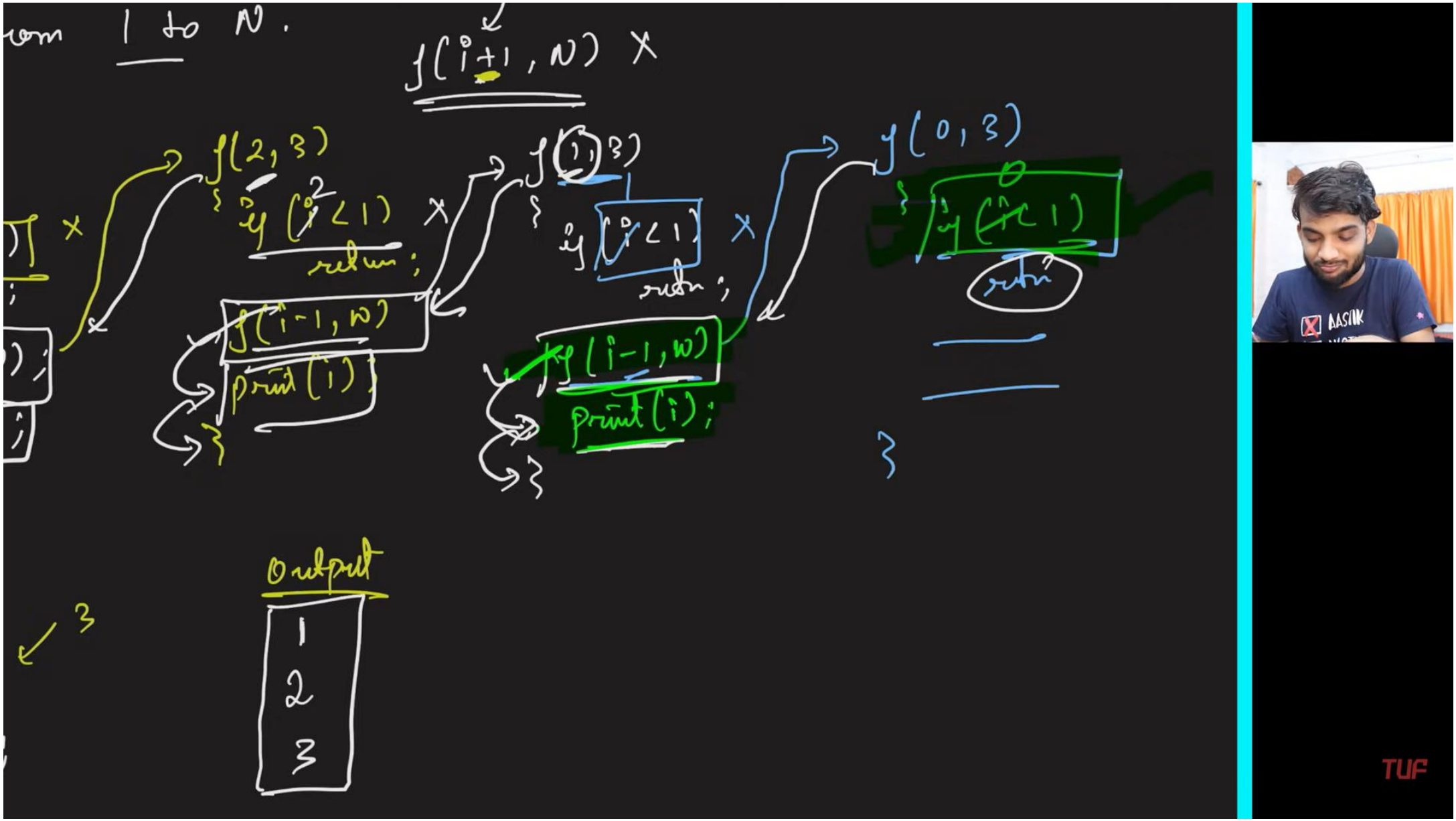
AASTIK


) Print from 1 to N.
f(i+1, N) X
f(3, 3)
if (i<1)
return;
f(i-1, N);
print(i);
f(2, 3)
f(1, 3)
f(0, 3)
main()
{
input(n)
f(n, n);
Output
1
2
3
TUF


) Print from 1 to N.
f(i+1, N) X
f(3, 3)
if (i<1)
return;
f(i-1, N);
print(i);
f(2, 3)
if (i<1)
return;
f(i-1, N)
print(i);
f(1, 3)
if (i<1)
return;
f(i-1, N)
print(i);
f(0, 3)
main()
{
input(n)
f(n, N);
Output
1
2
3
TUF


) Print from 1 to N.
f(i+1, N) X
f(3, 3)
if (i<1)
return;
f(i-1, N);
print(i);
f(2, 3)
f(1, 3)
f(0, 3)
main()
{
input(n)
f(3, N);
Output
1
2
3


) Print from 1 to N.
f(i+1, N) X
f(3, 3)
if (i<1)
return;
f(i-1, N);
print(i);
f(2, 3)
f(1, 3)
f(0, 3)
main()
{
input(n)
f(3, N);
Output
1
2
3


rom 1 to N.
f(i+1, N) X
f(2, 3)
f(1, 3)
f(0, 3)
if (i<1)
return;
f(i-1, N)
print(i);
Output
1
2
3

(8) Print from 1 to N.

f(i+1, N) X

f(3, 3)
if (3 < 1)
return;
f(i-1, N);
print(i);

f(2, 3)
if (2 < 1) X
return;
f(i-1, N)
print(i);

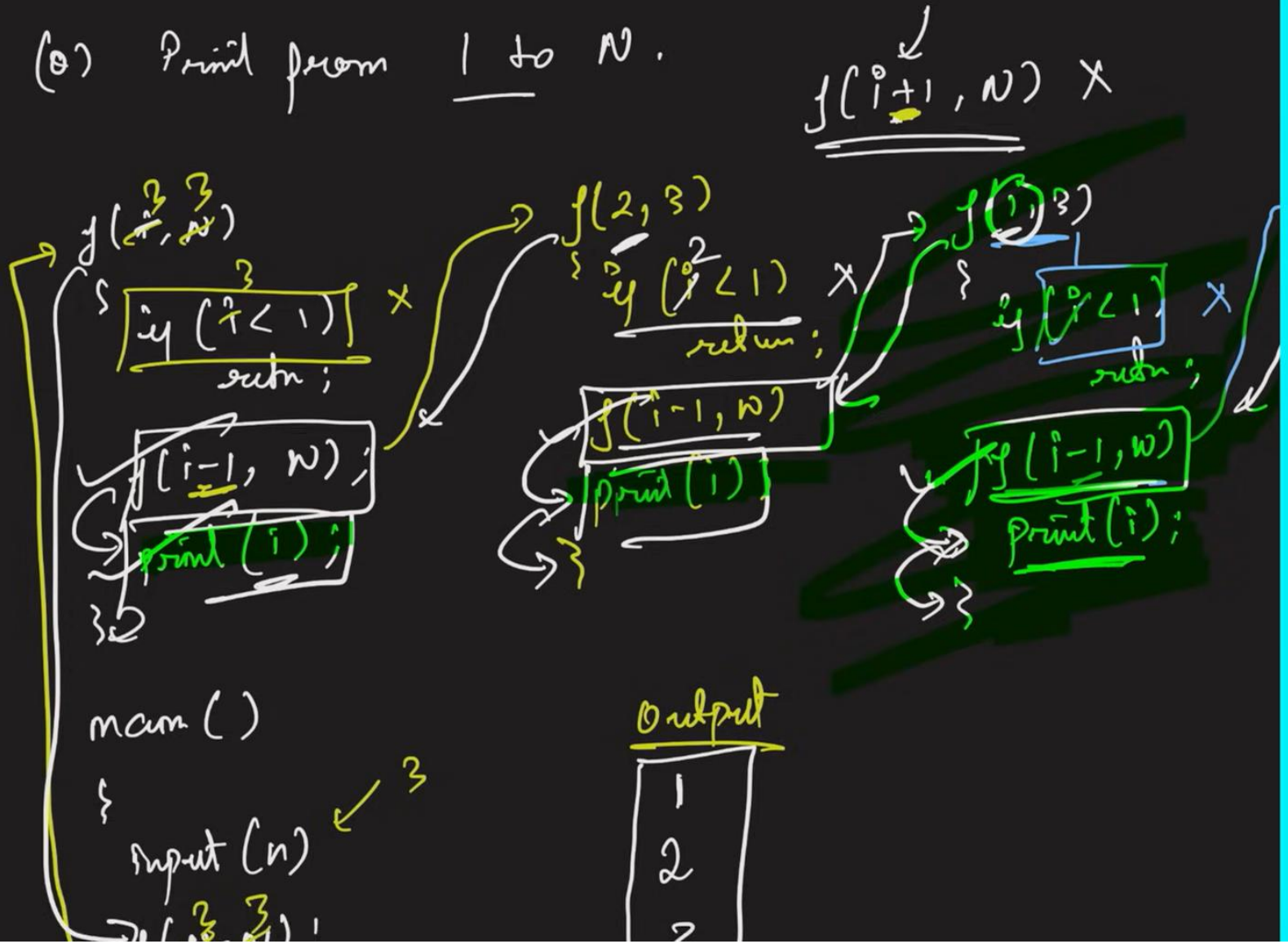
f(1, 3)
if (1 < 1) X
return;
f(i-1, N)
print(i);

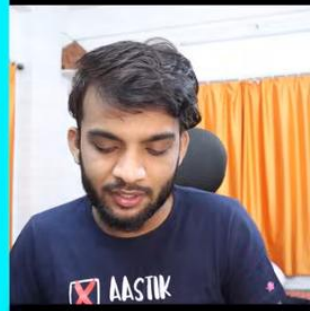
main()
{
input(n)
f(3, N);
}

Output
1
2
3

TUF

(8) Print from 1 to N.





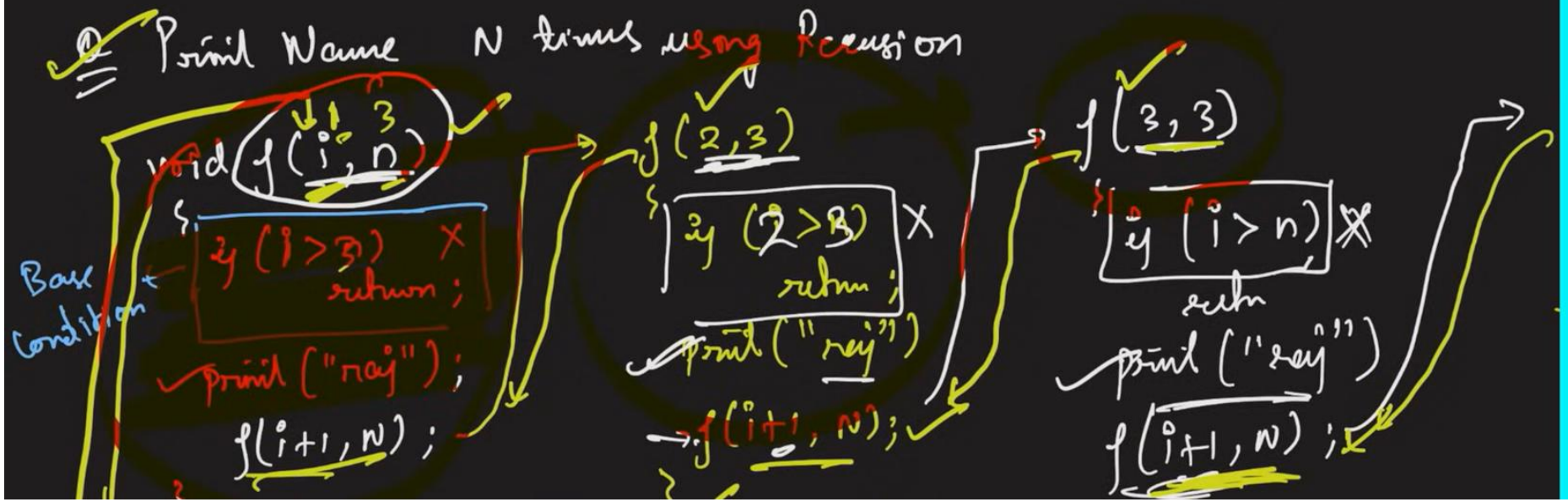
TUF

Print from N to 1.

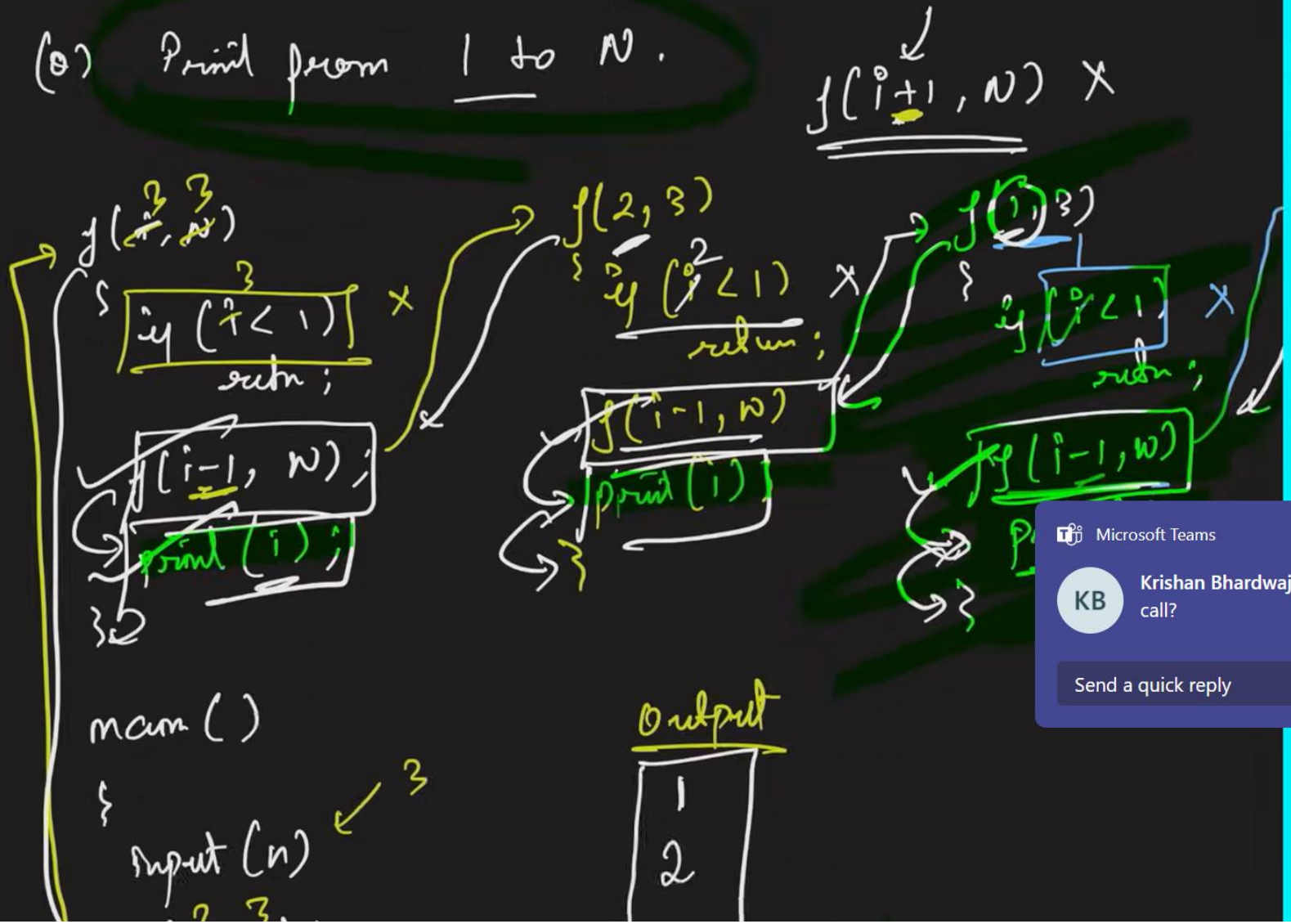
→ Print Linearly from 1 to N
(But By Backtrack)

→ Print from N to 1
(By Backtrack)

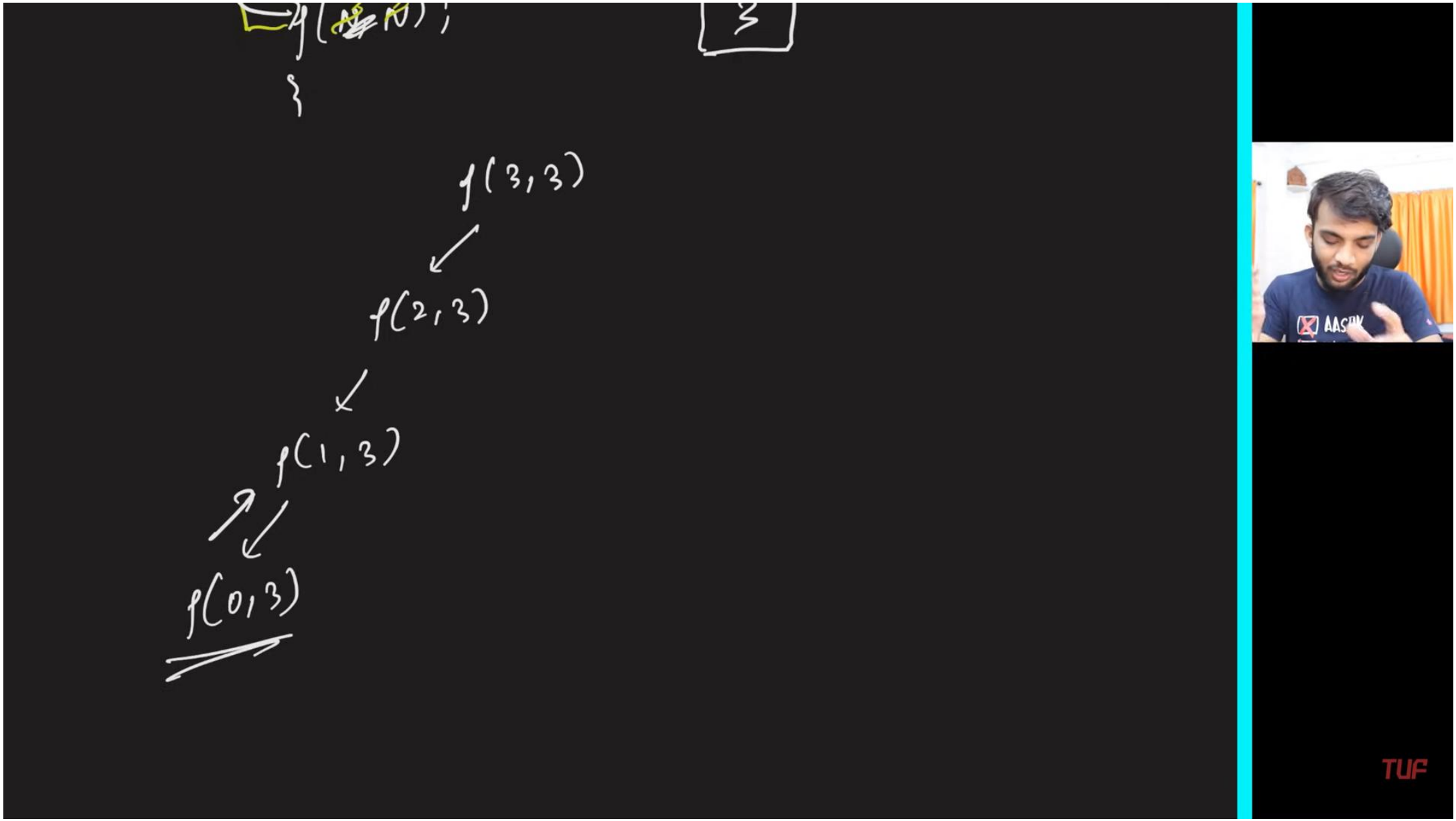
Q Print Name N times using Recursion

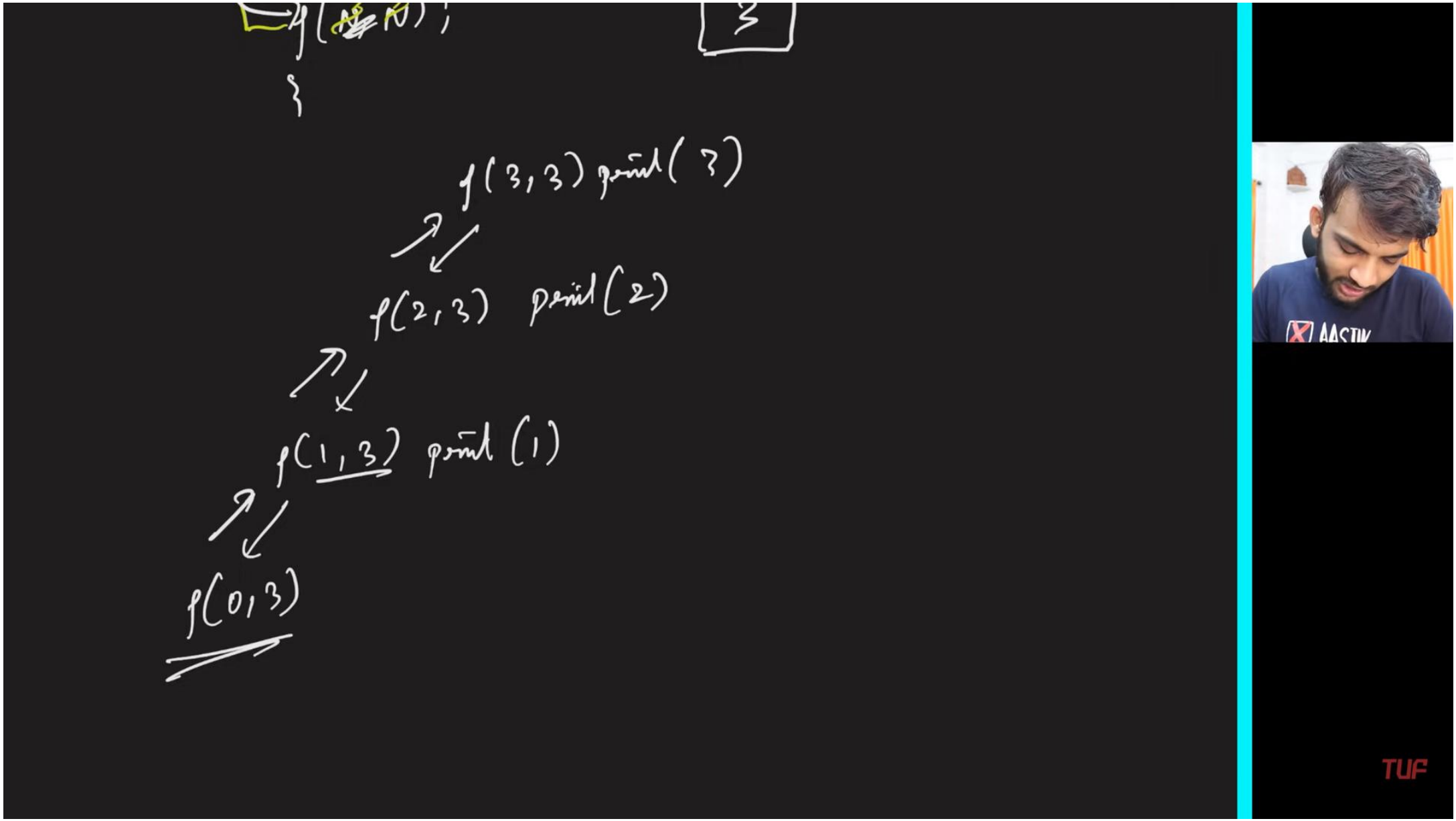


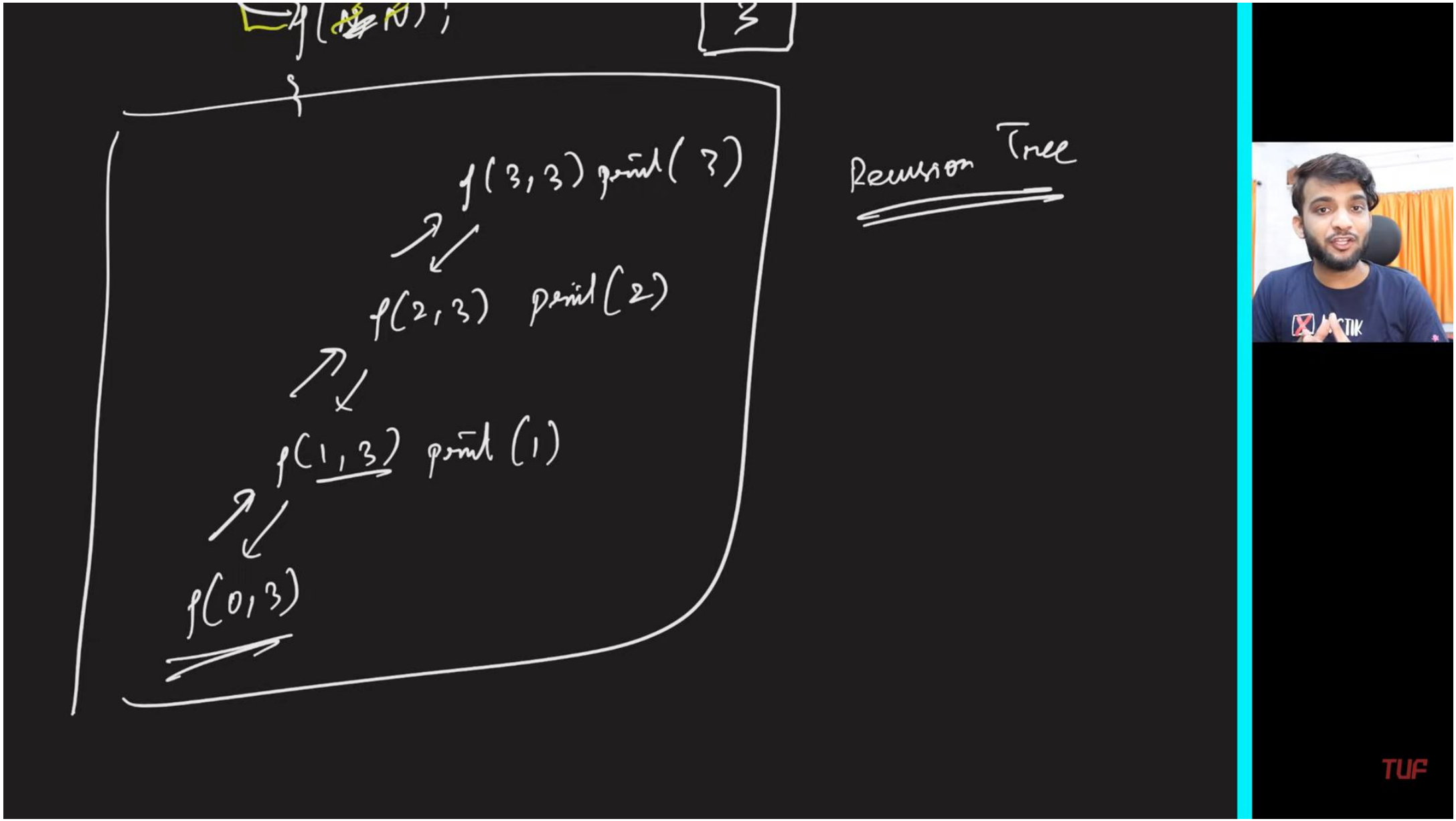
(8) Print from 1 to N.



TUF


f(3,3)
f(2,3)
f(1,3)
f(0,3)
3


f(3,3) print(3)
f(2,3) print(2)
f(1,3) print(1)
f(0,3)


f(3,3) print(3)
f(2,3) print(2)
f(1,3) print(1)
f(0,3)
Recursion Tree