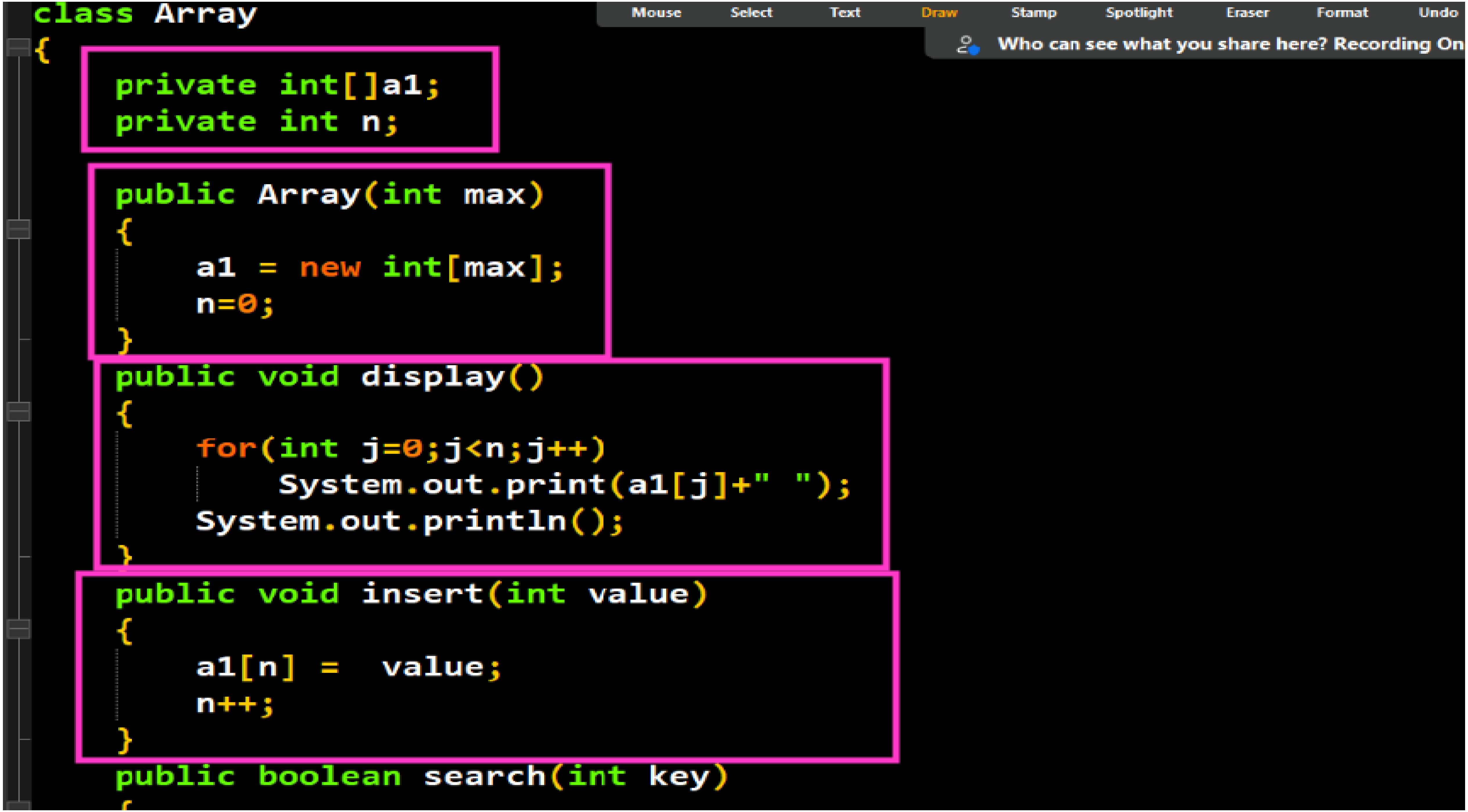
Sep22 : Day 3

Kiran Waghmare

CDAC Mumbai



Who can see what you share here? Recording On

```
class Array
{
    private int[]a1;
    private int n;

    public Array(int max)
    {
        a1 = new int[max];
        n=0;
    }
    public void display()
    {
        for(int j=0;j<n;j++)
            System.out.print(a1[j]+" ");
        System.out.println();
    }
    public void insert(int value)
    {
        a1[n] =  value;
        n++;
    }
    public boolean search(int key)
    {
```

Problem statement: Find duplicates in an array

- Given an array a1[] of size N which contains elements from 0 to N-1, you need to find all the elements occurring more than once in the given array.

- **Example 1:**
  - Input:
    - N = 4
    - a[] = {0,3,1,2}
  - Output: -1
  - Explanation: N=4 and all elements from 0 to (N-1 = 3) are present in the given array. Therefore output is -1.

- **Example 2:**
  - Input:
    - N = 5
    - a[] = {2,3,1,2,3}
  - Output: 2 3
  - Explanation: 2 and 3 occur more than once in the given array.

Problem statement: Removing punctuations from a given string

- **Given a string, remove the punctuation from the string if the given character is a punctuation character, as classified by the current C locale. The default C locale classifies these characters as punctuation:**
  - ! " # $ % & ' ( ) * + , - . / : ; ? @ [\ ] ^ _ ` { | } ~

- **Example 1:**
  - Input : %welcome' to @cdacmumbai?<s
  - Output : welcome to cdacmumbai

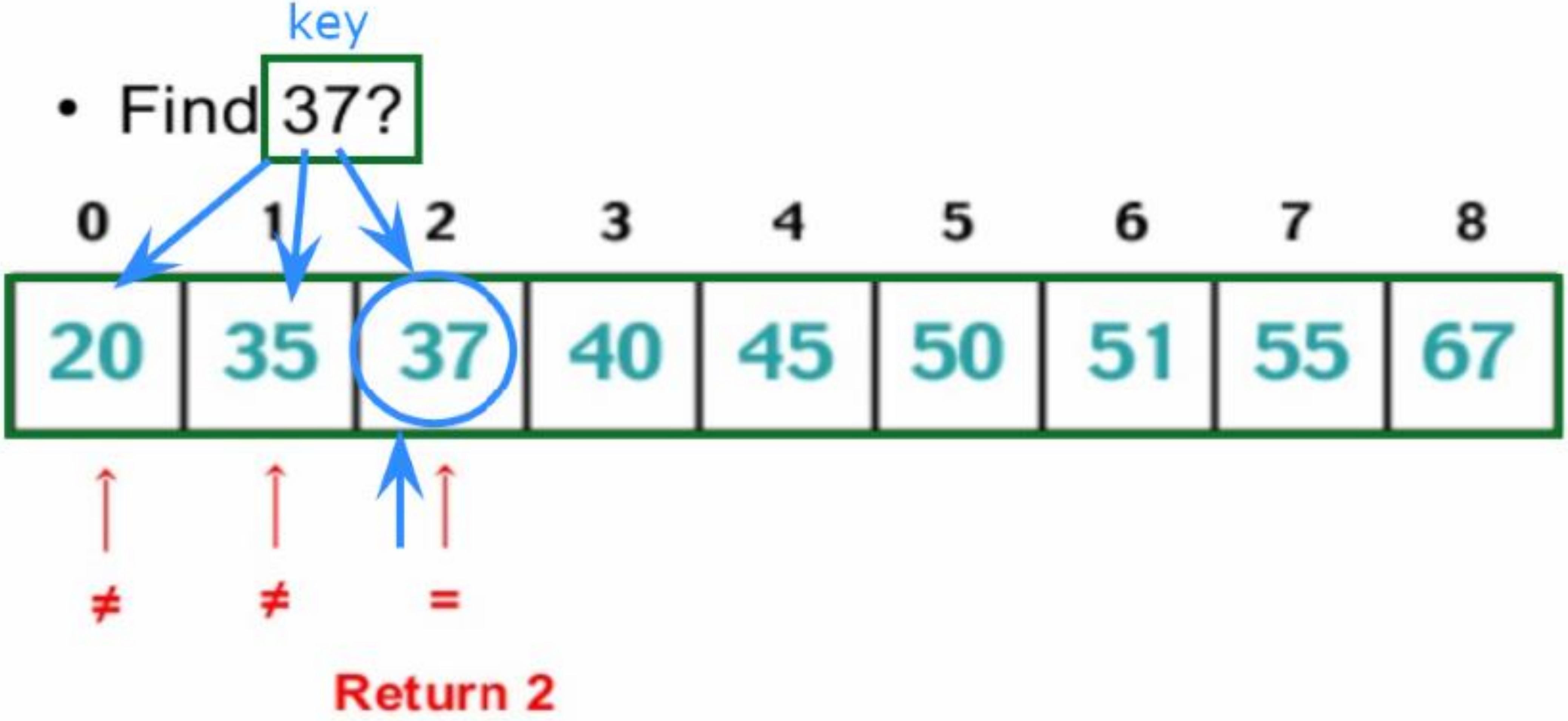
- **Example 2:**
  - Input : Hello!!!, he said ---and went**.
  - Output : Hello he said and went

Problem statement: Program to find the initials of a name.

- Given a string name, we have to find the initials of the name
- Examples 1:
  - Input : Kabhi Haa Kabhi Naa
  - Output : K H K N
    - We take the first letter of all
    - words and print in capital letter.
- Example 2:
  - Input : Mahatma Gandhi
  - Output : M G
- Example 3:
  - Input : Shah Rukh Khan
  - Output : S R K

  - Example 4: your own name

Linear Search

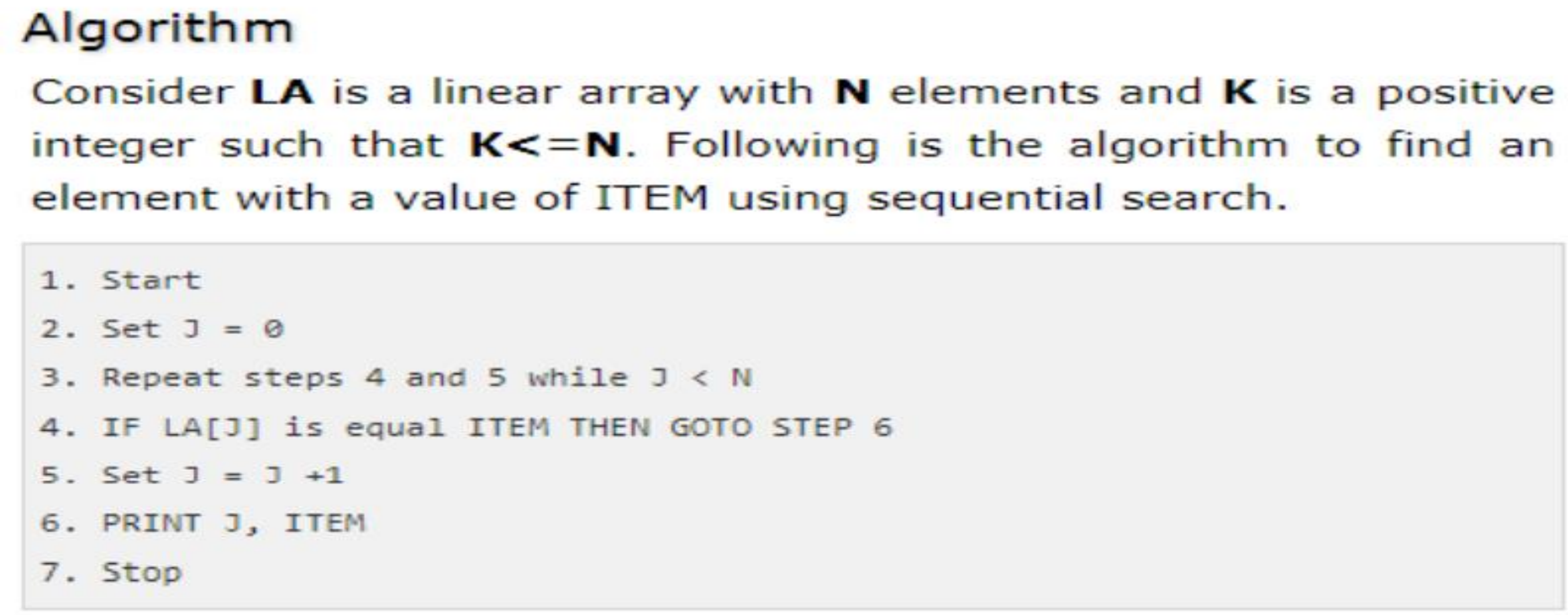


- Find 37?

0	1	2	3	4	5	6	7	8
20	35	37	40	45	50	51	55	67

Return 2

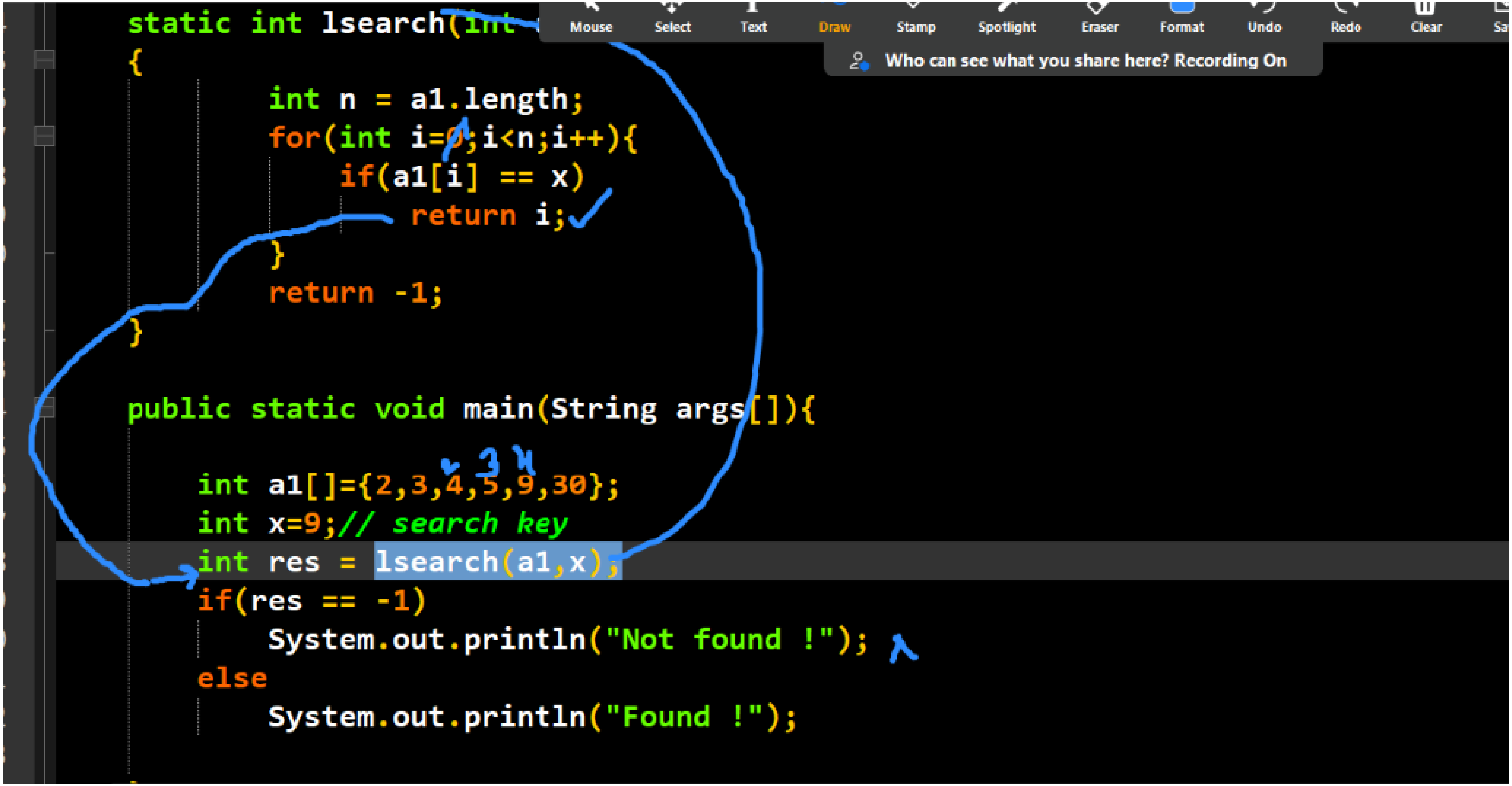
Linear Search



Algorithm

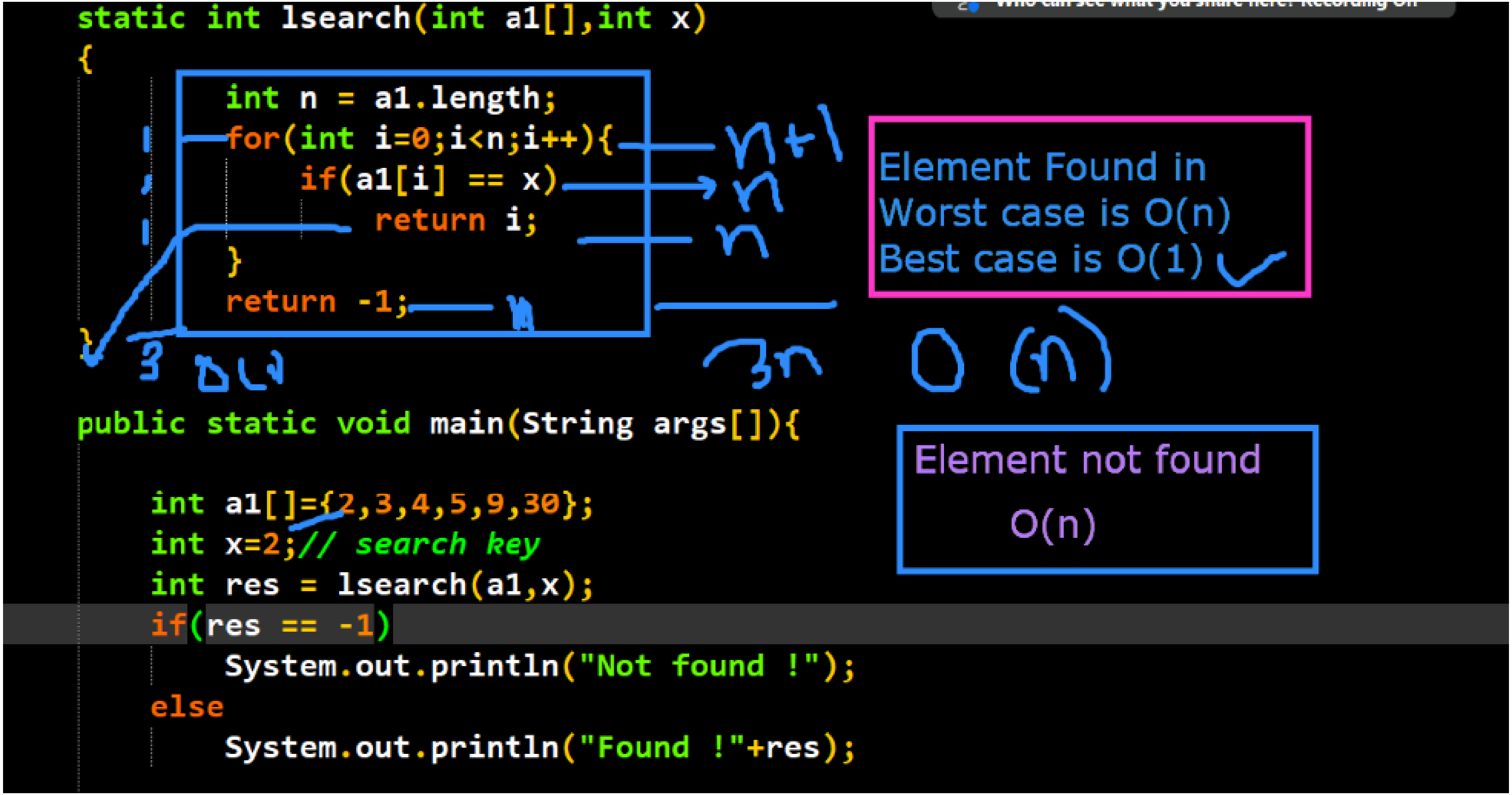
Consider **LA** is a linear array with **N** elements and **K** is a positive integer such that **K<=N**. Following is the algorithm to find an element with a value of ITEM using sequential search.

```
1. Start
2. Set J = 0
3. Repeat steps 4 and 5 while J < N
4. IF LA[J] is equal ITEM THEN GOTO STEP 6
5. Set J = J +1
6. PRINT J, ITEM
7. Stop
```



Who can see what you share here? Recording On

```
static int lsearch(int
{
        int n = a1.length;
        for(int i=0;i<n;i++){
            if(a1[i] == x)
                return i;
        }
        return -1;
}

public static void main(String args[]){

    int a1[]={2,3,4,5,9,30};
    int x=9;// search key
    int res = lsearch(a1,x);
    if(res == -1)
        System.out.println("Not found !");
    else
        System.out.println("Found !");
```



```
static int lsearch(int a1[],int x)
{
        int n = a1.length;
        for(int i=0;i<n;i++){
            if(a1[i] == x)
                return i;
        }
        return -1;
}

public static void main(String args[]){

    int a1[]={2,3,4,5,9,30};
    int x=2;// search key
    int res = lsearch(a1,x);
    if(res == -1)
        System.out.println("Not found !");
    else
        System.out.println("Found !"+res);
```

Element Found in
Worst case is O(n)
Best case is O(1) ✓

Element not found

O(n)

Program 3

Problem: Given an array arr[] of n elements, write a function to search a given element x in arr[].

Examples :

Input : arr[] = {10, 20, 80, 30, 60, 50, 110, 100, 130, 170}
x = 110;
Output : 6
Element x is present at index 6

Input : arr[] = {10, 20, 80, 30, 60, 50,
110, 100, 130, 170}
x = 175;
Output : -1
Element x is not present in arr[].

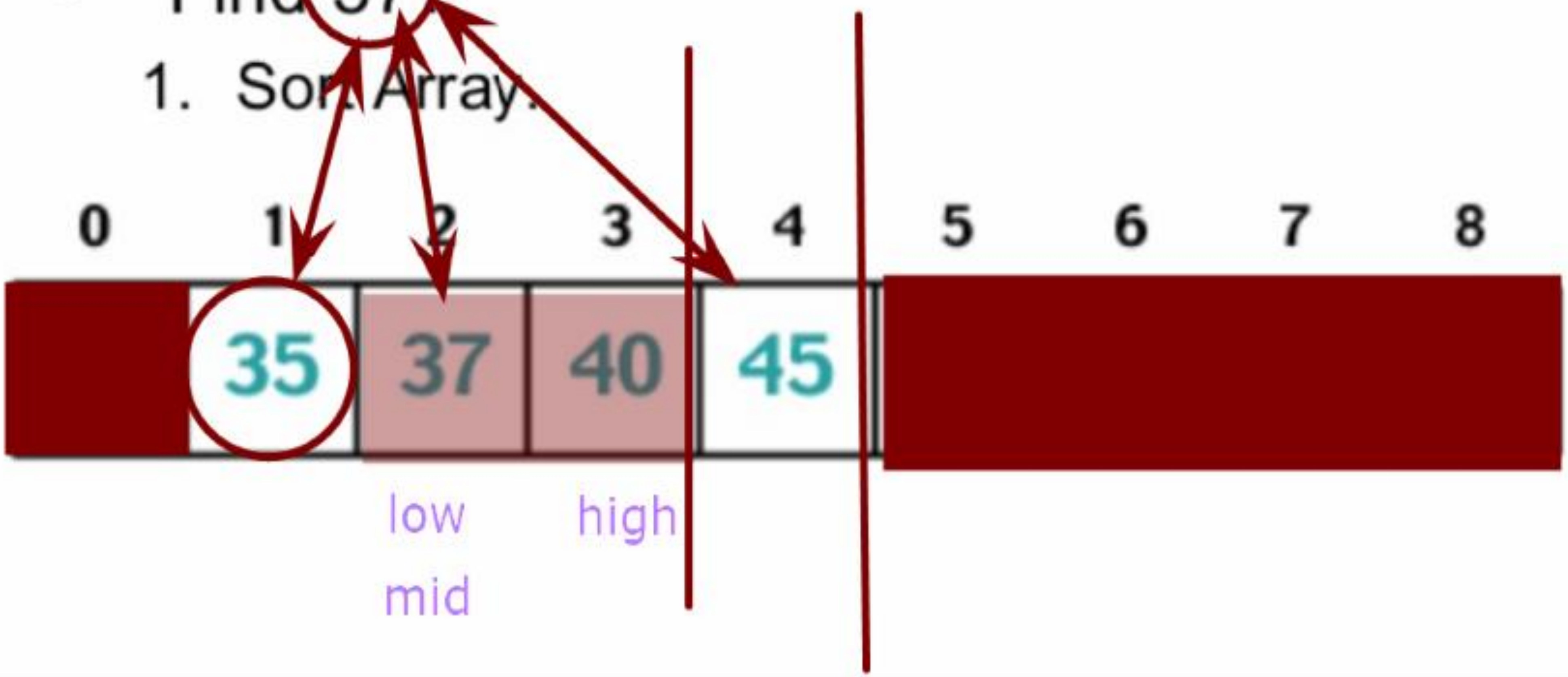
-Sorted order

2

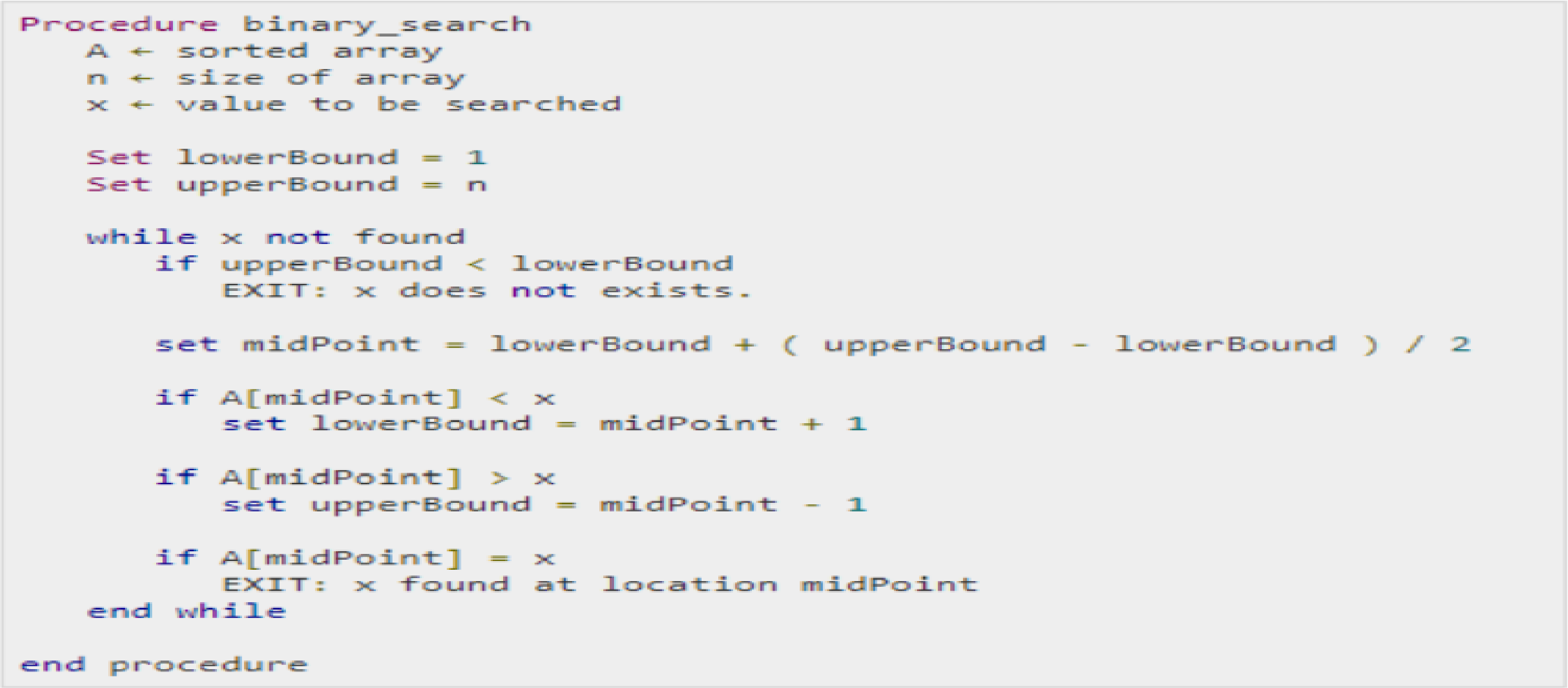
Binary Search

mid=(low+high)/2
=(0+8)/2
=4

- Find 37?
  1. Sort Array.



Binary Search



```
Procedure binary_search
   A ← sorted array
   n ← size of array
   x ← value to be searched

   Set lowerBound = 1
   Set upperBound = n

   while x not found
      if upperBound < lowerBound
         EXIT: x does not exists.

      set midPoint = lowerBound + ( upperBound - lowerBound ) / 2

      if A[midPoint] < x
         set lowerBound = midPoint + 1

      if A[midPoint] > x
         set upperBound = midPoint - 1

      if A[midPoint] = x
         EXIT: x found at location midPoint
   end while

end procedure
```

-Sorted order

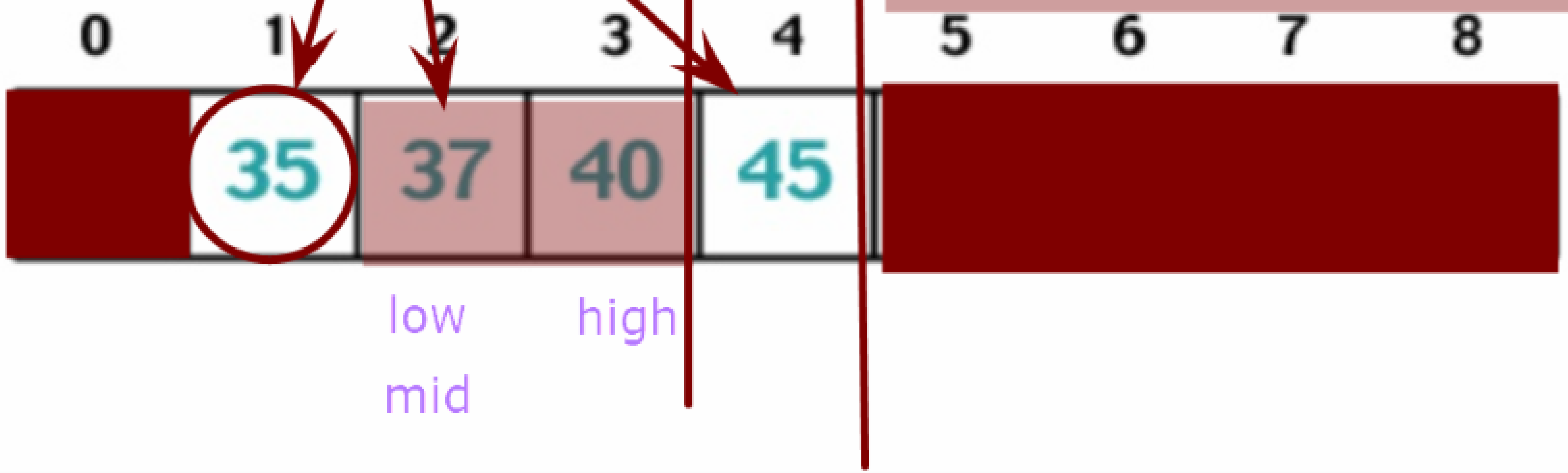
2

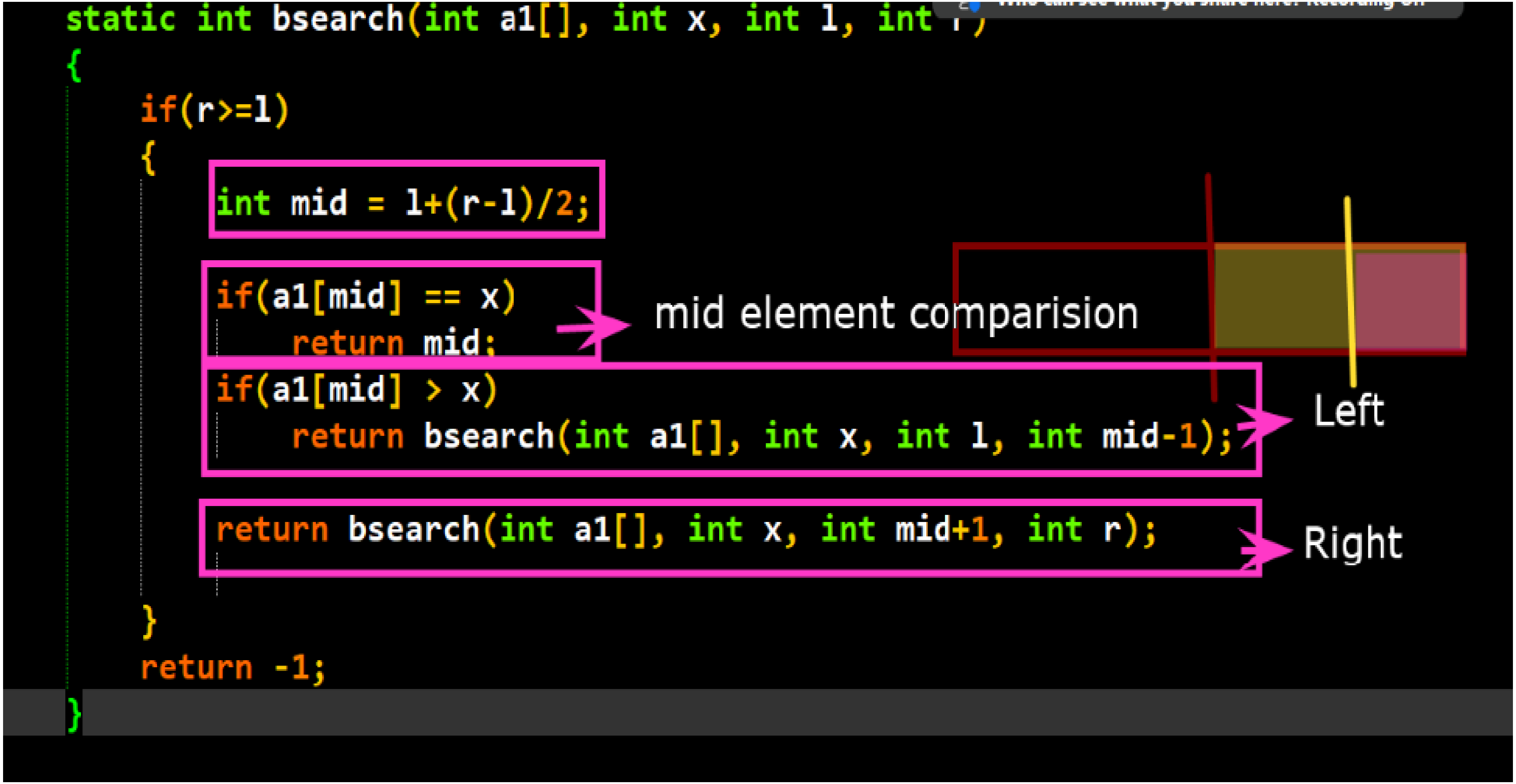
Binary Search

mid=(low+high)/2
=(0+8)/2
=4

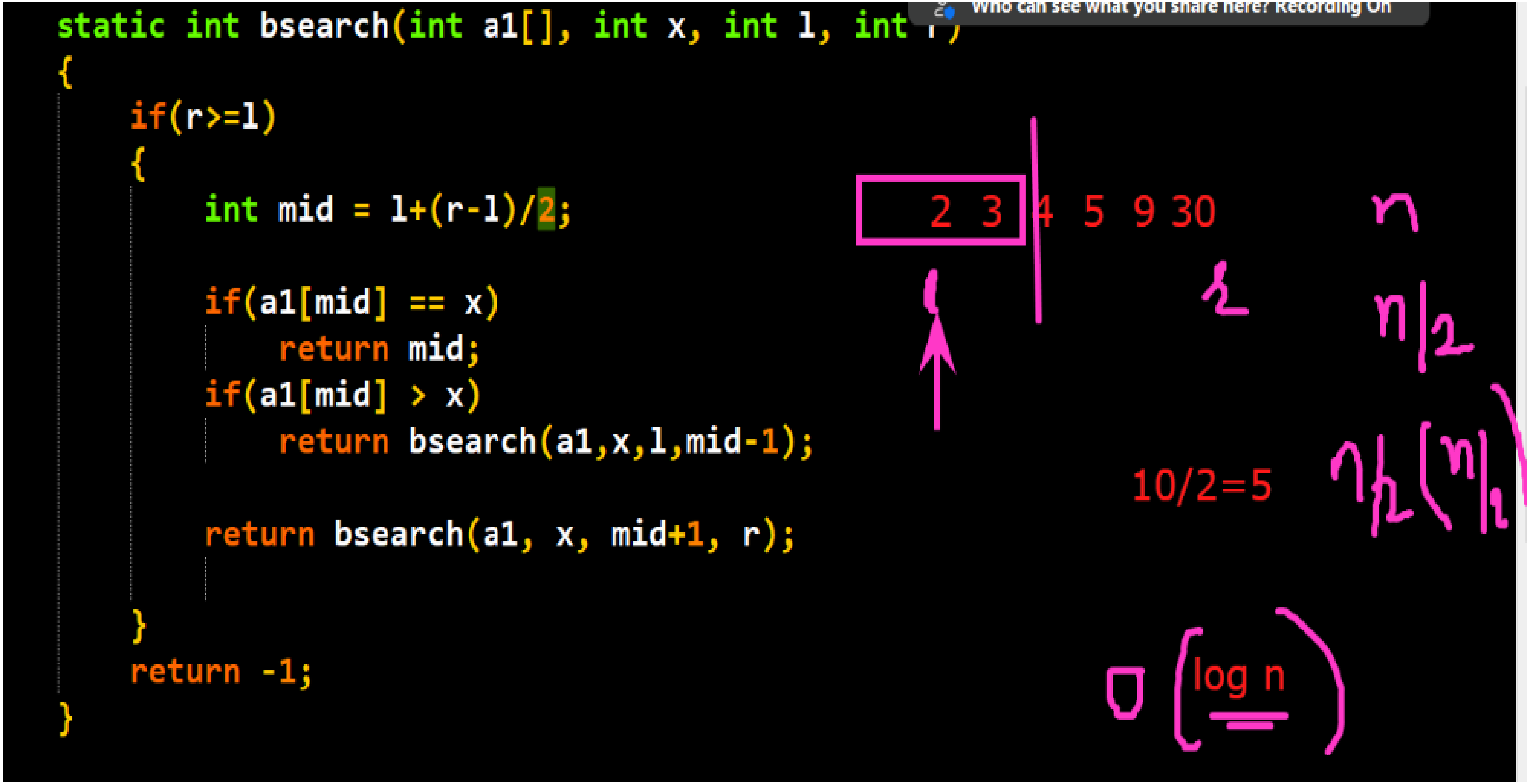
- Find 37?
  1. Sort Array:

Not found=> stop when low>high





```
static int bsearch(int a1[], int x, int l, int r)
{
    if(r>=l)
    {
        int mid = l+(r-l)/2;

        if(a1[mid] == x)
            return mid;
        if(a1[mid] > x)
            return bsearch(int a1[], int x, int l, int mid-1);

        return bsearch(int a1[], int x, int mid+1, int r);

    }
    return -1;
}
```

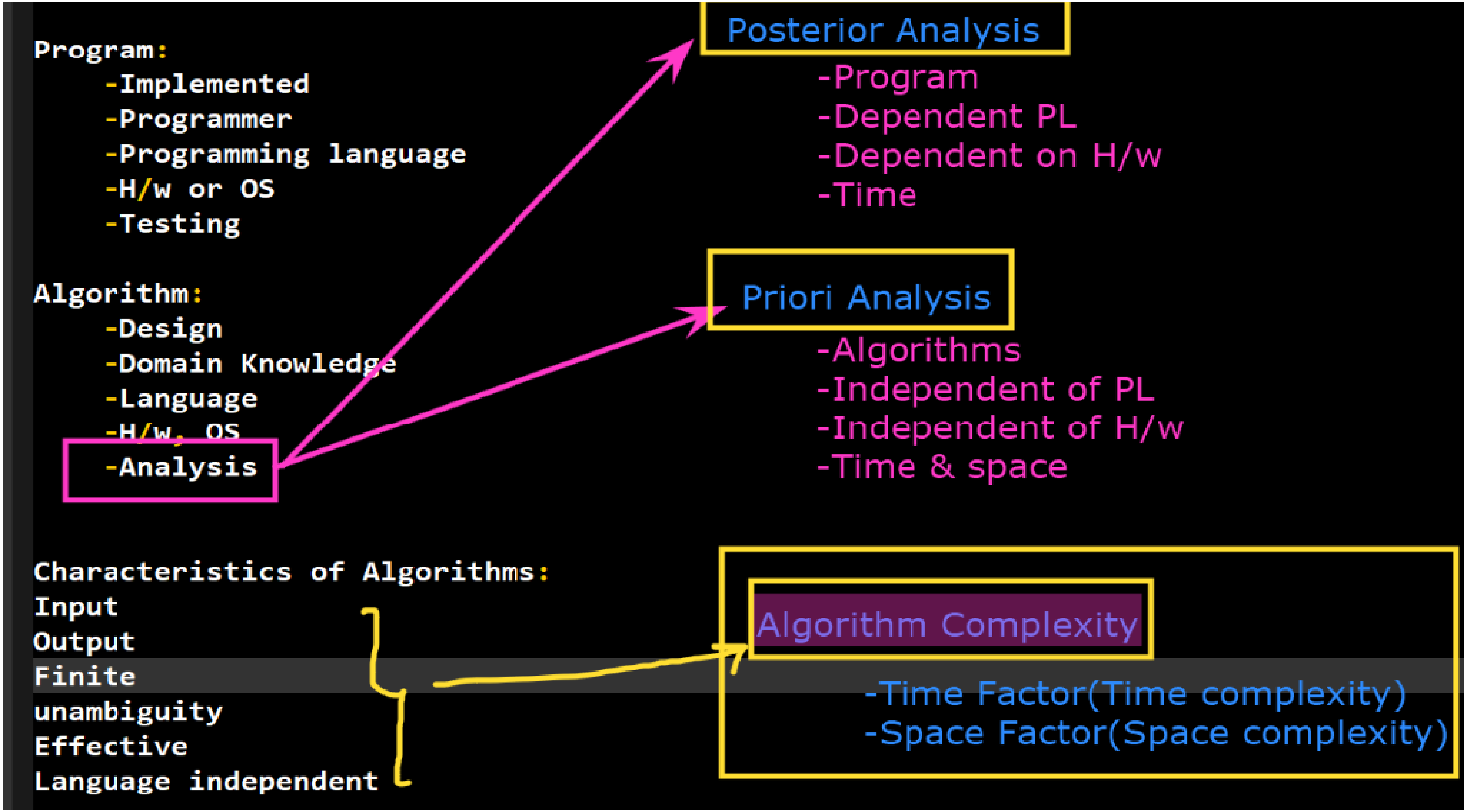


Who can see what you share here? Recording On

```
static int bsearch(int a1[], int x, int l, int r)
{
    if(r>=l)
    {
        int mid = l+(r-l)/2;

        if(a1[mid] == x)
            return mid;
        if(a1[mid] > x)
            return bsearch(a1,x,l,mid-1);

        return bsearch(a1, x, mid+1, r);

    }
    return -1;
}
```

Algorithms & Data Structure Complexity

Kiran Waghmare



Program:

- -Implemented
- -Programmer
- -Programming language
- -H/w or OS
- -Testing

Algorithm:

- -Design
- -Domain Knowledge
- -Language
- -H/w, OS
- -Analysis

Posterior Analysis

- -Program
- -Dependent PL
- -Dependent on H/w
- -Time

Priori Analysis

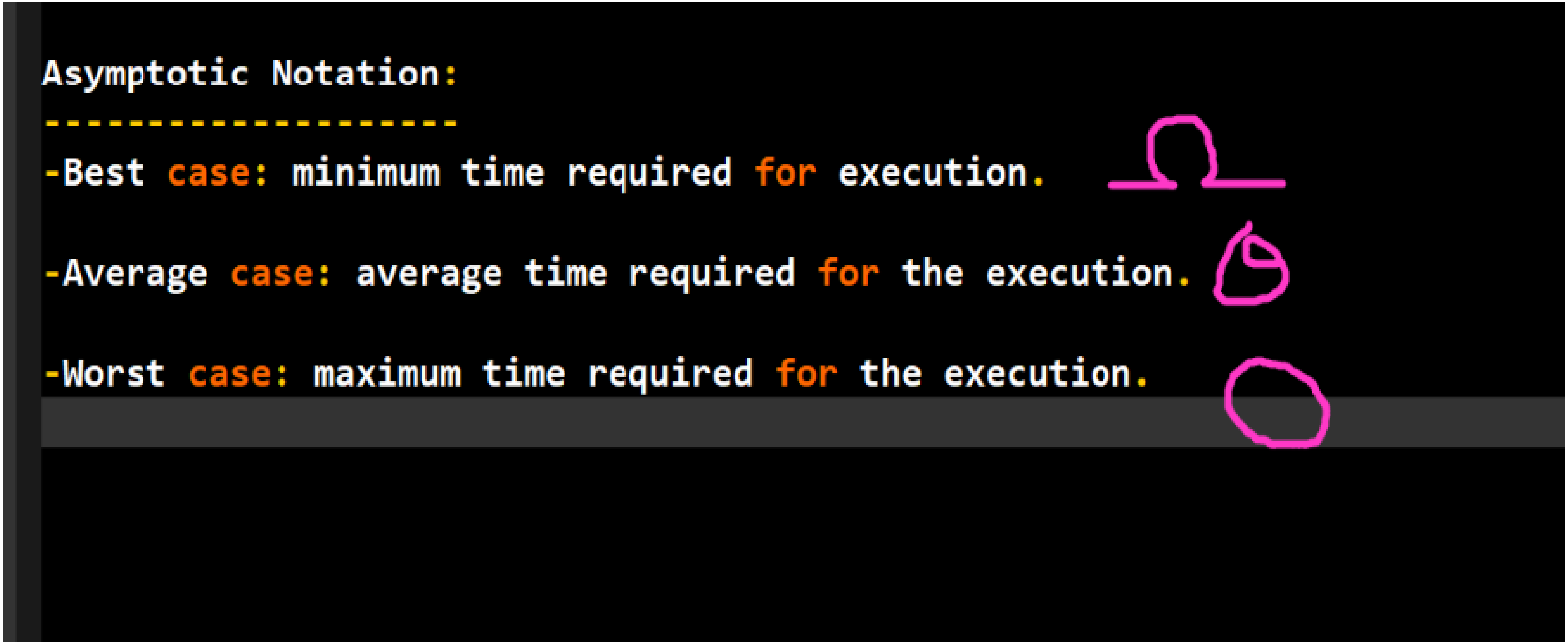
- -Algorithms
- -Independent of PL
- -Independent of H/w
- -Time & space

Characteristics of Algorithms:

Input
Output
Finite
unambiguity
Effective
Language independent

Algorithm Complexity

- -Time Factor(Time complexity)
- -Space Factor(Space complexity)



```
Asymptotic Notation:
--------------------
-Best case: minimum time required for execution.              Ω

-Average case: average time required for the execution.        Θ

-Worst case: maximum time required for the execution.          O
```

```
if(x==5)            →    1
{
    stmt;           →    1
}
```

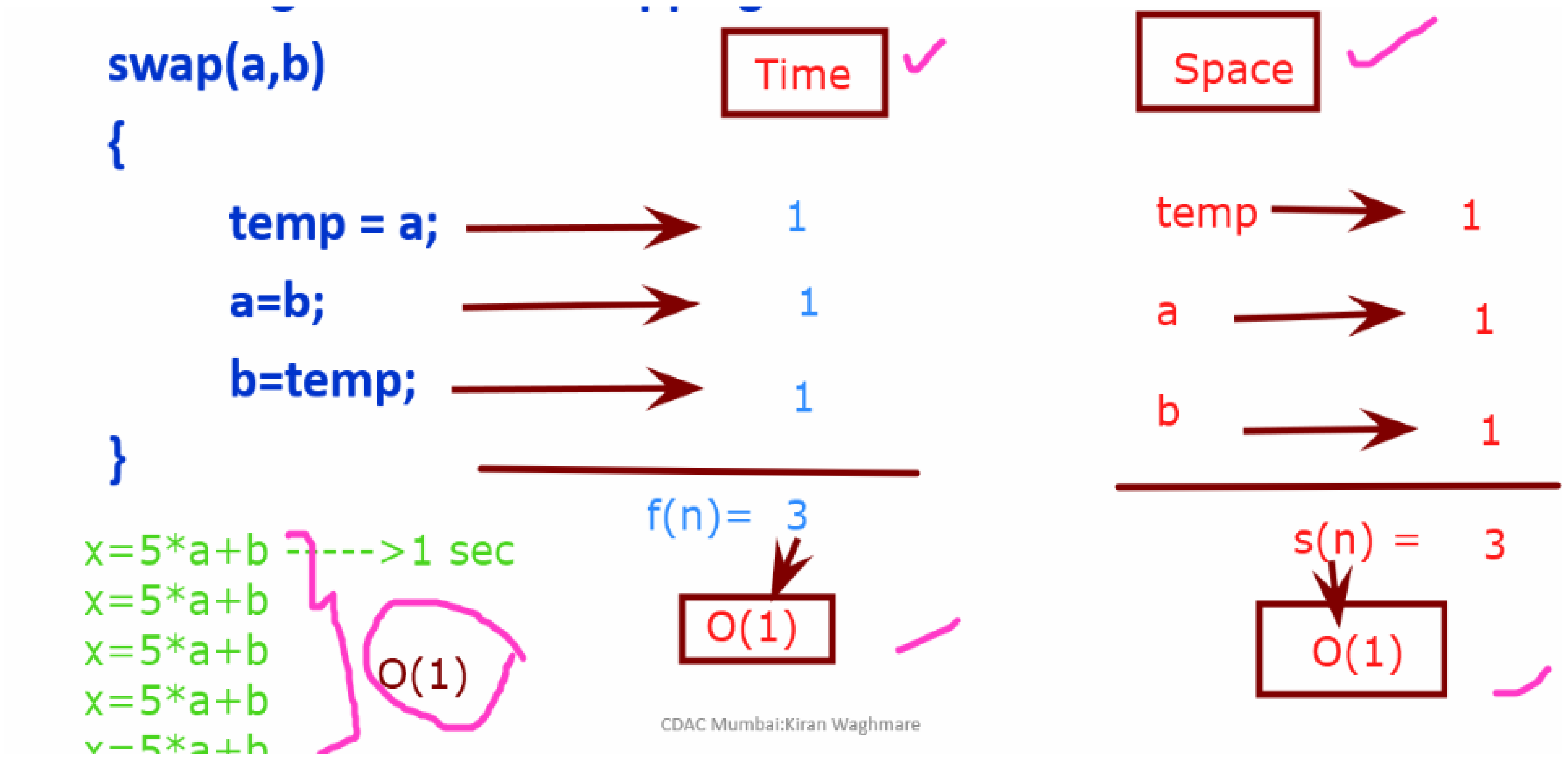
f(n) = 2 sec

$\boxed{O(2)} \rightarrow \text{constant} \leftarrow O(1)$

```
if(x==5)
{
    stmt;
}
```

```
for(int i=0;i<5;)            →    6          i=0,1,2,3,4,5
{
    SOP("done");             →    5
    i++;                     →    5
}                                ____
                                  16
```

```
for(int i=0;i<n;i++)         →    n+1
{
    SOP("done");             }    n
}
```



```
swap(a,b)
{
      temp = a;
      a=b;
      b=temp;
}
```

Time ✓

temp = a; → 1

a=b; → 1

b=temp; → 1

f(n)= 3

$\boxed{O(1)}$

Space ✓

temp → 1

a → 1

b → 1

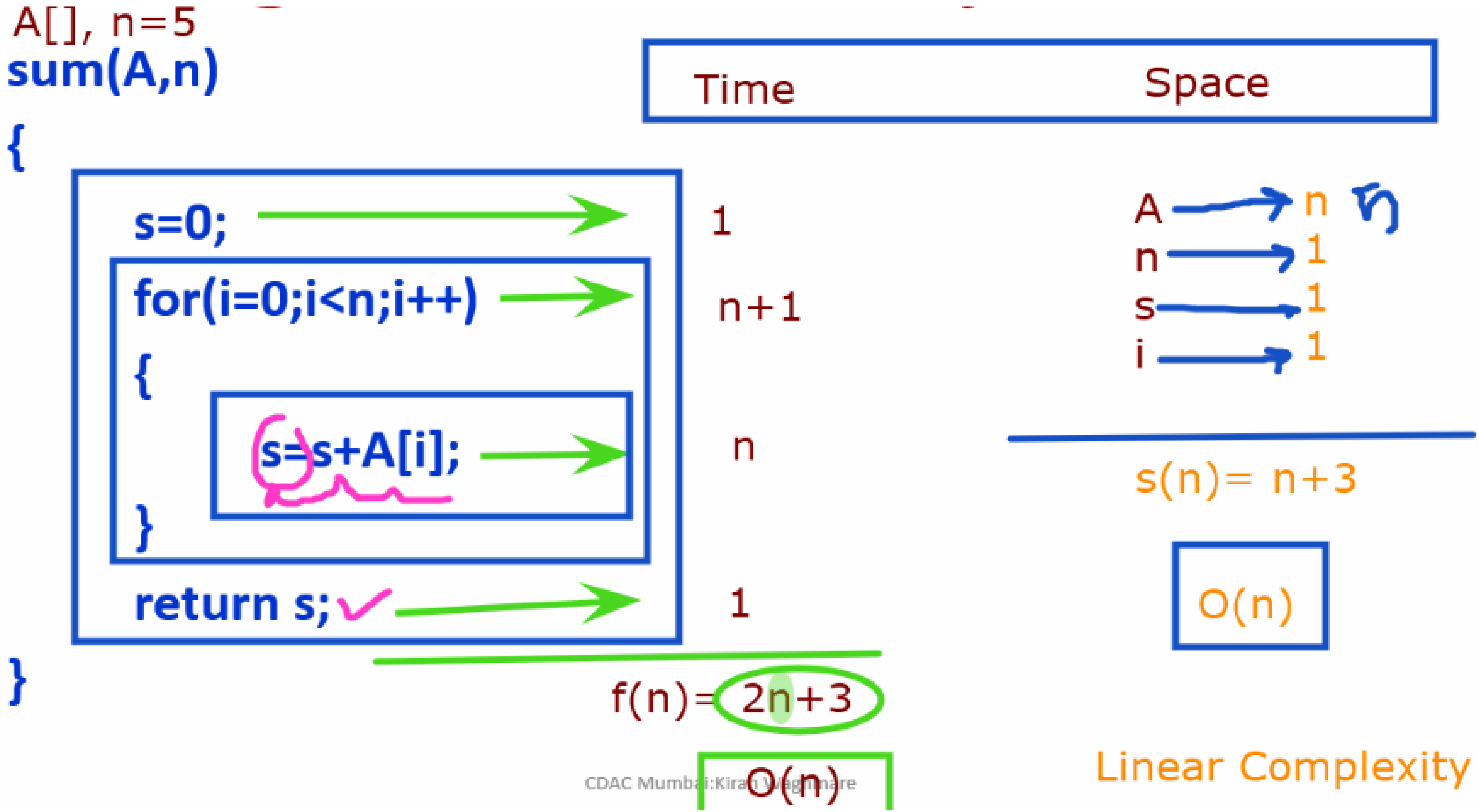
s(n) = 3

$\boxed{O(1)}$

x=5*a+b ----->1 sec
x=5*a+b
x=5*a+b
x=5*a+b
x=5*a+b

O(1)

CDAC Mumbai:Kiran Waghmare

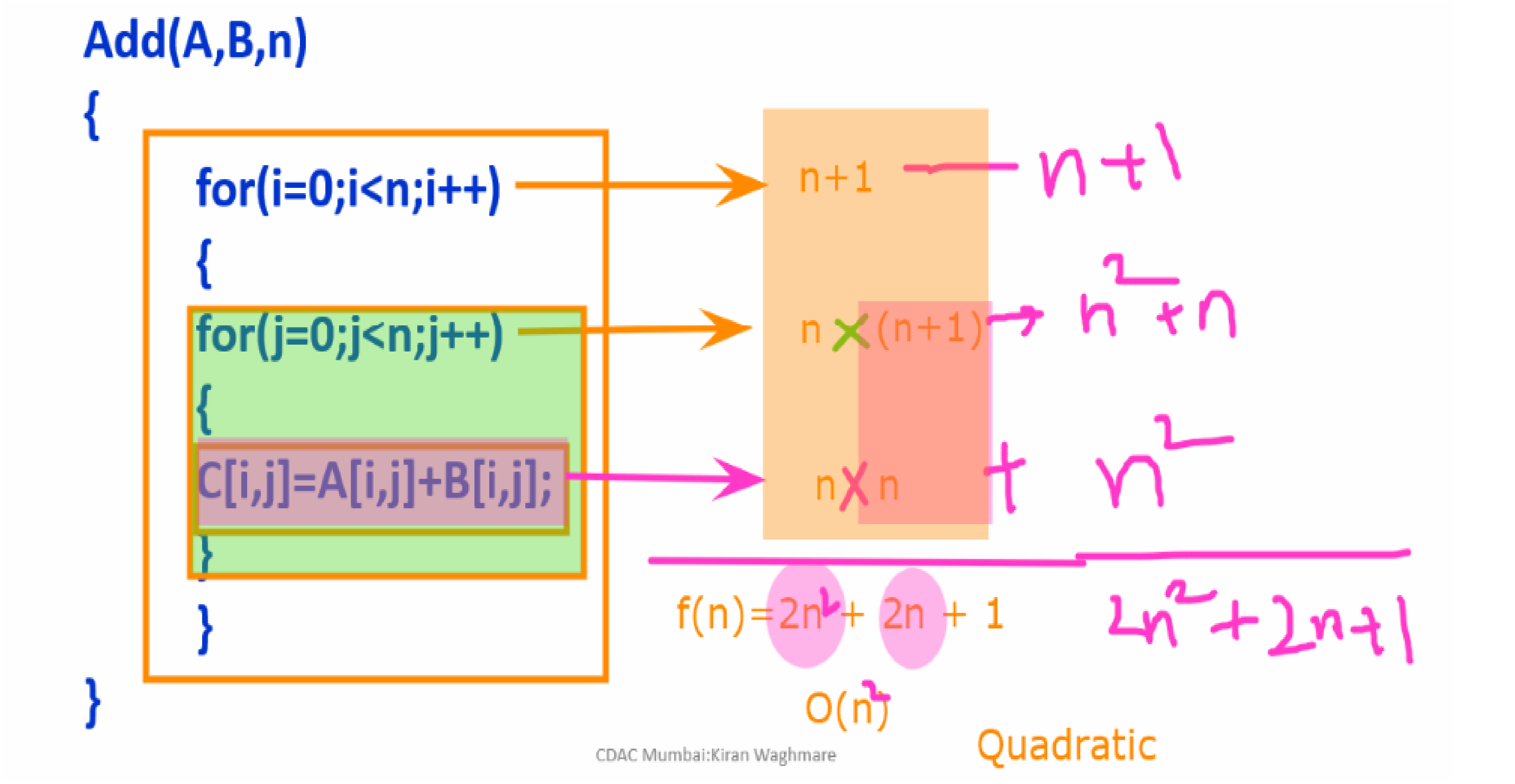


A[], n=5

```
sum(A,n)
{
    s=0;
    for(i=0;i<n;i++)
    {
        s=s+A[i];
    }
    return s;
}
```

Time	Space
1	A → n
n+1	n → 1
n	s → 1
1	i → 1
f(n)= 2n+3	s(n)= n+3
O(n)	O(n)

Linear Complexity



```
Add(A,B,n)
{
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            C[i,j]=A[i,j]+B[i,j];
        }
    }
}
```

$n+1$ — $n+1$

$n \times (n+1) \rightarrow n^2+n$

$n \times n$ $+ \; n^2$

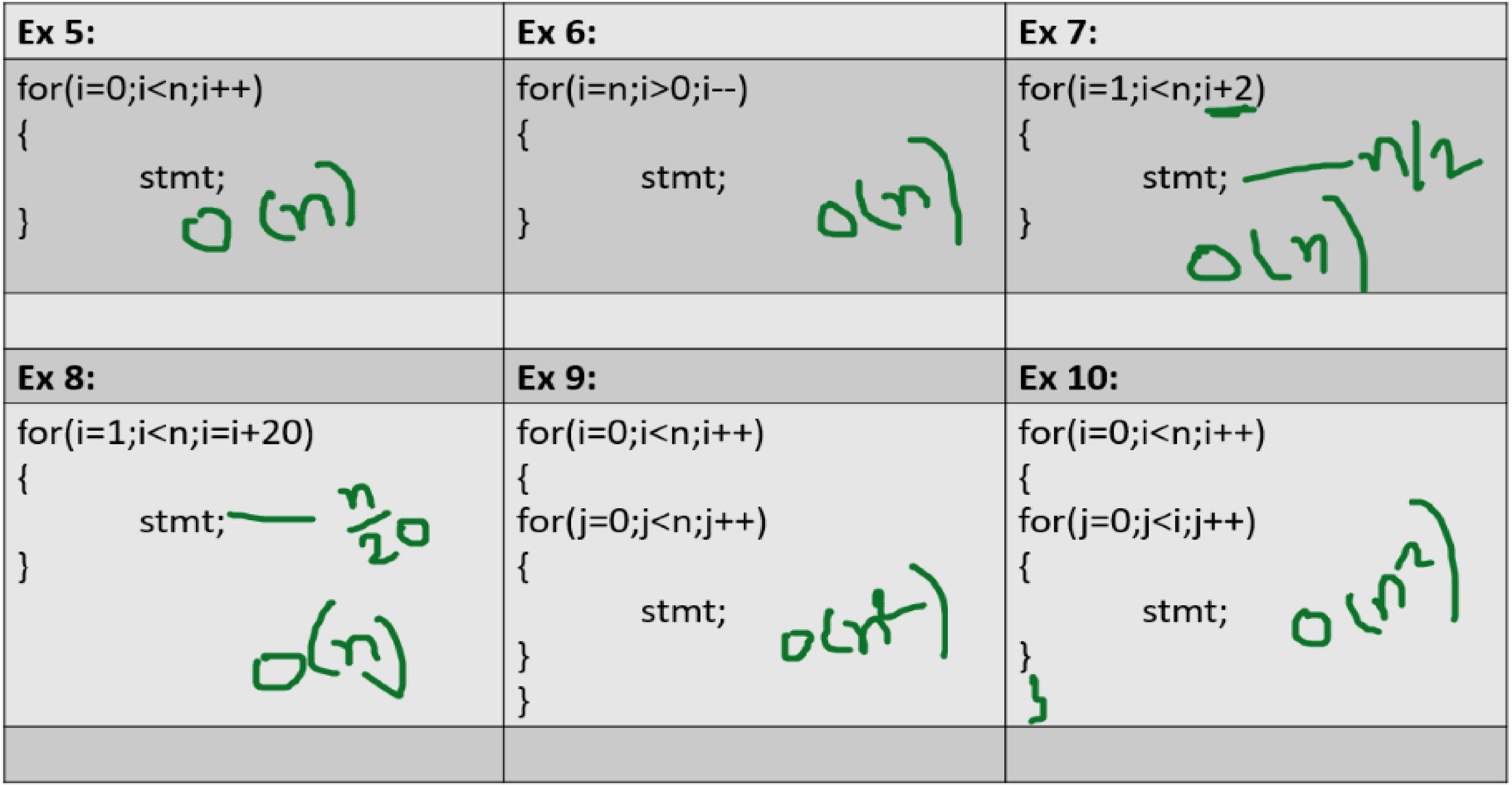
$2n^2+2n+1$

$f(n)=2n^2+2n+1$

$O(n^2)$

Quadratic

CDAC Mumbai:Kiran Waghmare



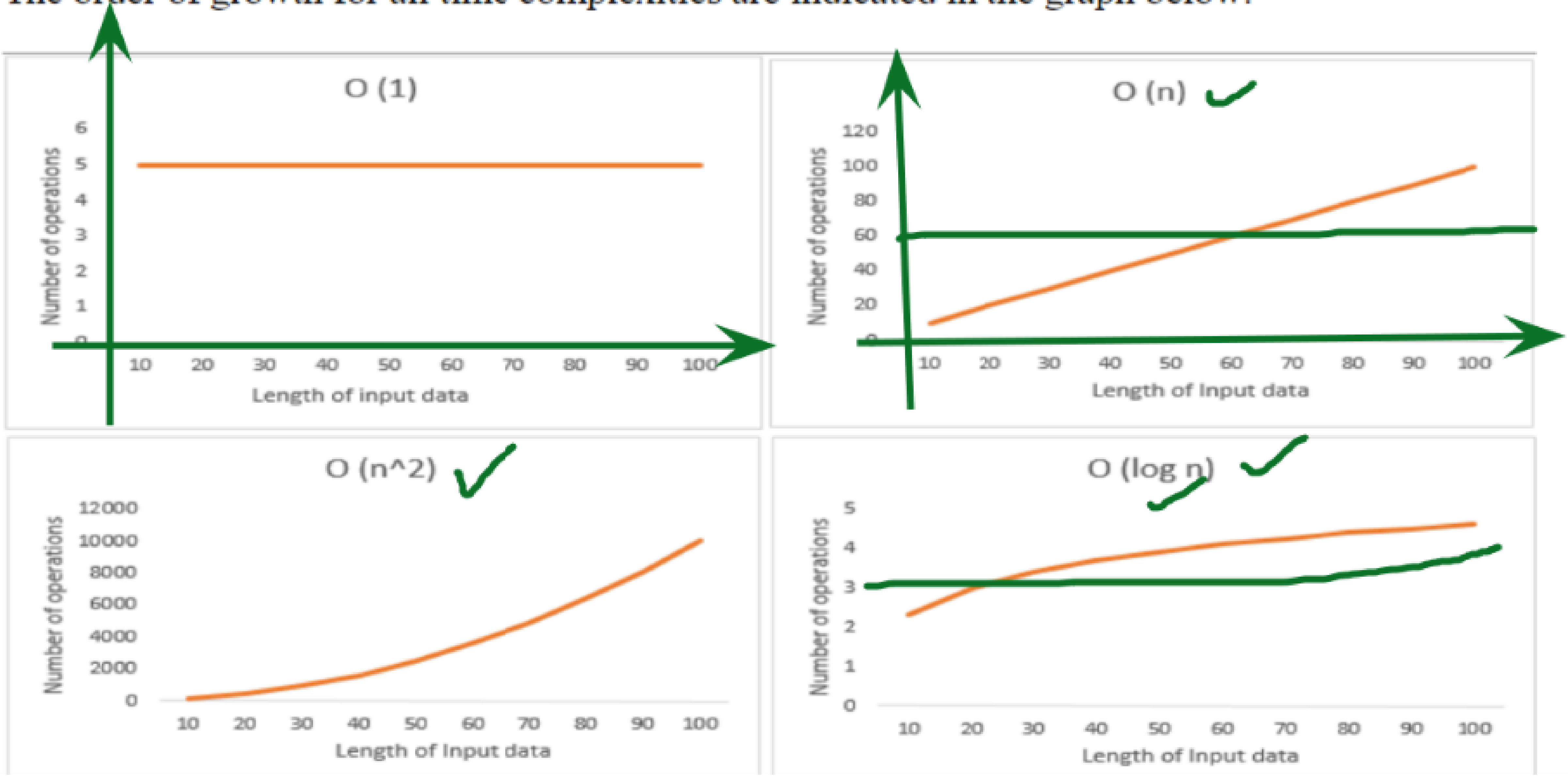
Ex 5:	Ex 6:	Ex 7:

for(i=0;i<n;i++)
{
stmt;
}
$O(n)$
	for(i=n;i>0;i--)
	{
	stmt;
	}
	$O(n)$
		for(i=1;i<n;i+2)
		{
		stmt; — $n/2$
		}
		$O(n)$

Ex 8:	**Ex 9:**	**Ex 10:**

for(i=1;i<n;i=i+20)
{
stmt; — $\frac{n}{20}$
}
$O(n)$
	for(i=0;i<n;i++)
	{
	for(j=0;j<n;j++)
	{
	stmt;
	}
	}
	$O(n^2)$
		for(i=0;i<n;i++)
		{
		for(j=0;j<i;j++)
		{
		stmt;
		}
		}
		$O(n^2)$

The order of growth for all time complexities are indicated in the graph below:


O (1)
Number of operations
Length of input data
O (n)
Number of operations
Length of Input data
O (n^2)
Number of operations
Length of Input data
O (log n)
Number of operations
Length of Input data