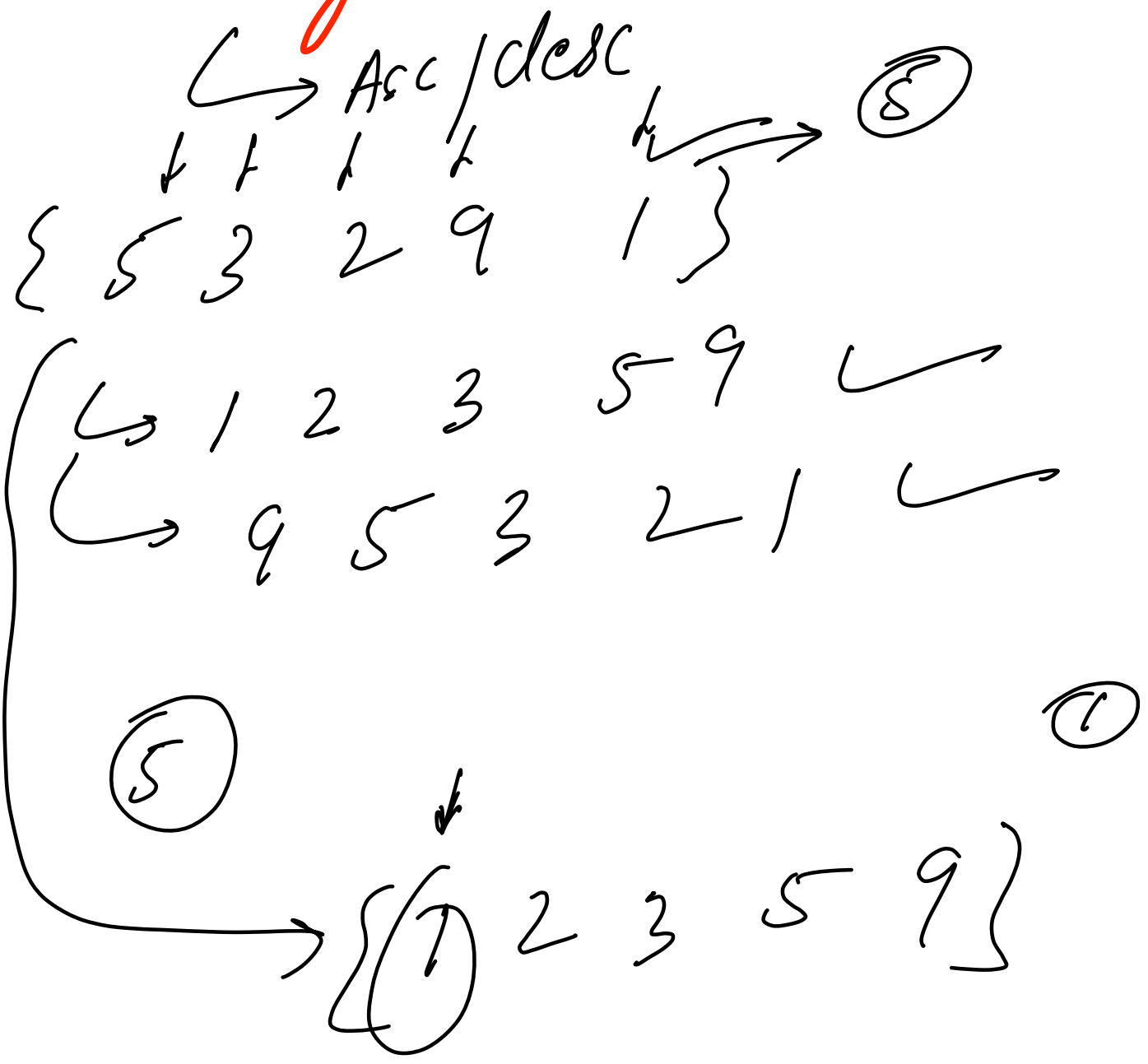
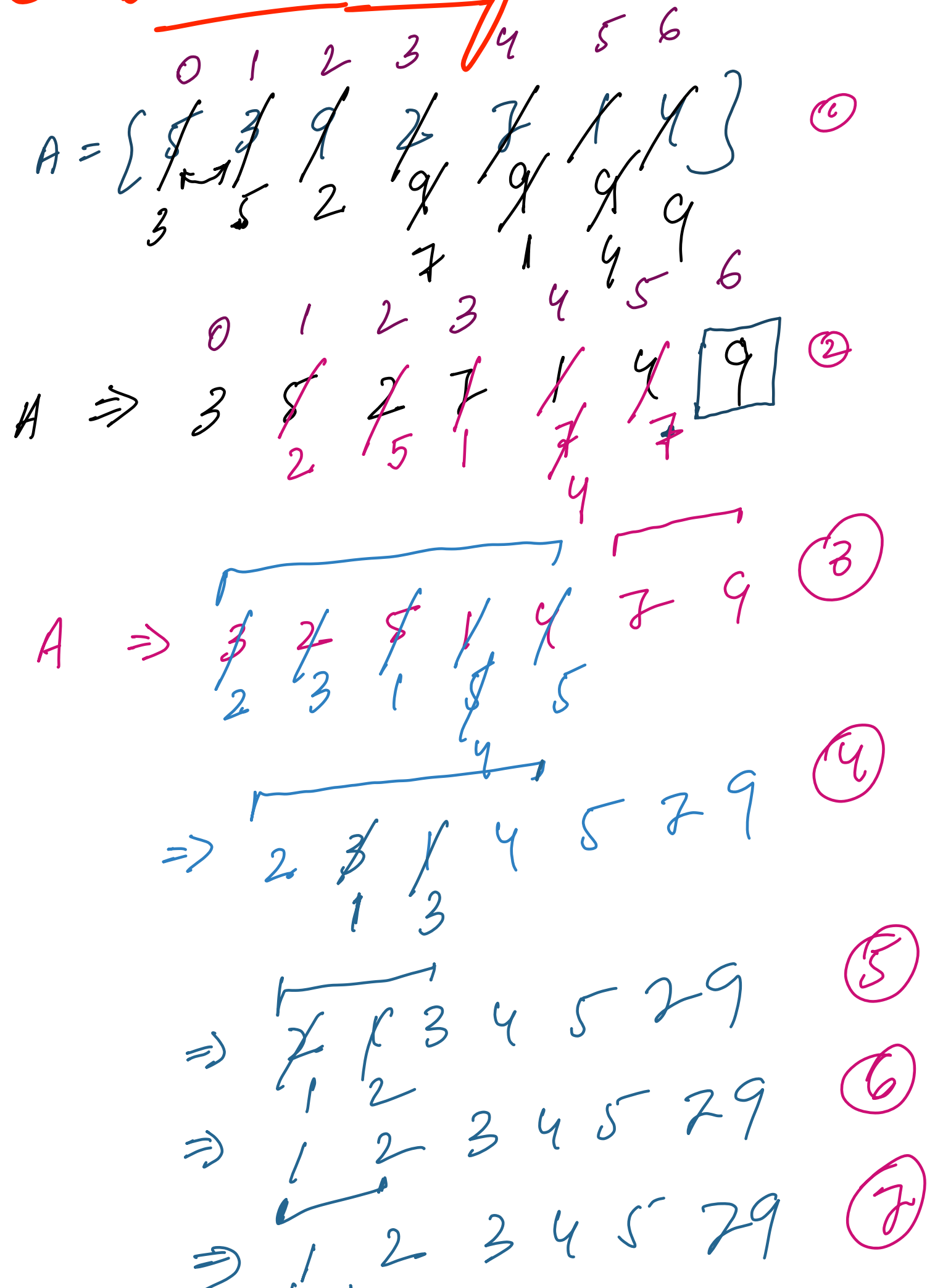
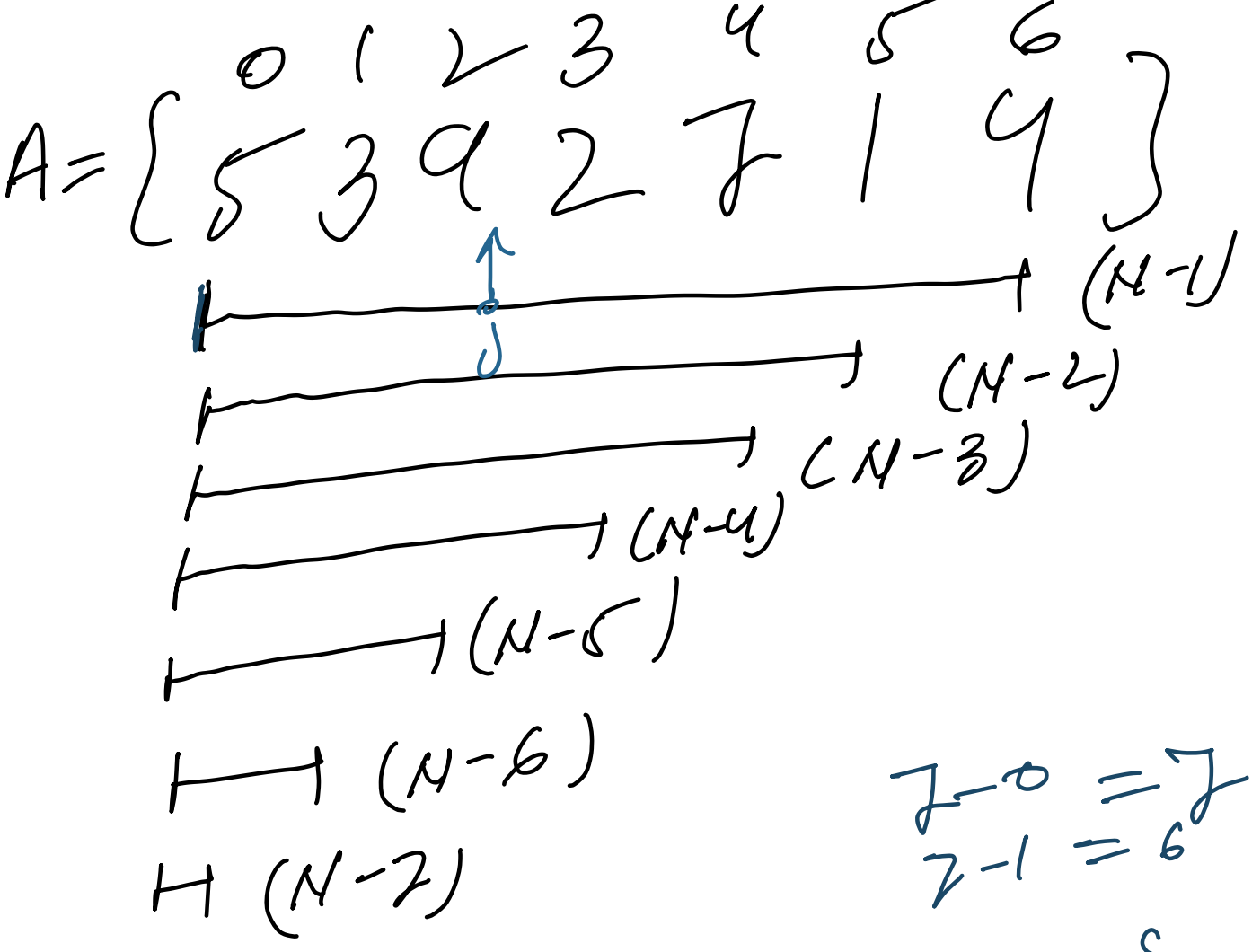
Sorting



- Bubble sort
- Selection sort
- Insertion
- **merge
- quick sort
- Heap sort

① Bubble Sorting





j-0 = j
j-1 = 6

```
for(i=0; i<N; i++){
    for(j=0; j<N-i; j++){
        if(A[j] < A[j-1])
            //swap
    }
}
```

$i=0 \rightarrow j=0$ 1 2 3 4 5 6 $(N-i)$ $(N-0)$

$i=1 \rightarrow j=0$ 1 2 3 4 5 $(N-1)$

$i=2 \rightarrow j=0$ 1 2 3 4 $(N-2)$

```
public class BubbleSort {
    public static void main(String[] args) {
        int [] A = {5,3,9,2,7,1,4} ;
        int N = A.length ;
        int k=2 ;

        for(int i=0 ; i<N ; i++){
            for(int j=0 ; j<N-i-1 ; j++){
                //Swapping
                if(A[j]>A[j+1]){
                    int temp = A[j] ;
                    A[j] = A[j+1] ;
                    A[j+1] = temp ;
                }
            }
        }


        for(int i=0 ; i<N ; i++){
            System.out.print(A[i] + " ");
        }
        System.out.println();
        System.out.println(A[N-k]);
    }
}
```

```
public class BubbleSort {
    public static void main(String[] args) {
        int [] A = {5,3,9,2,7,1,4} ;
        int N = A.length ;
        int k=2 ;

        for(int i=0 ; i<N ; i++){
            for(int j=0 ; j<N-i-1 ; j++){
                //Swapping
                if(A[j]>A[j+1]){
                    int temp = A[j] ;
                    A[j] = A[j+1] ;
                    A[j+1] = temp ;
                }
            }
            if ( i==k ) {
                break ;
            }
        }


        for(int i=0 ; i<N ; i++){
            System.out.print(A[i] + " ");
        }
        System.out.println();
        System.out.println(A[N-k]);
    }
}
```

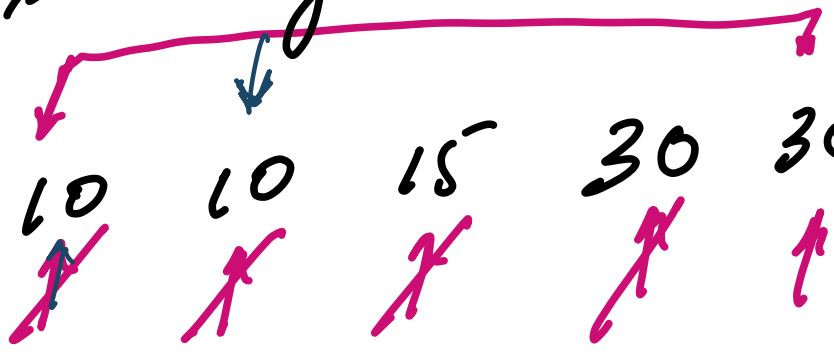
Q: find the Duplicate elements

A = { 30 10 15 10 40 30 }

o/p ⇒ 30
10

Soln

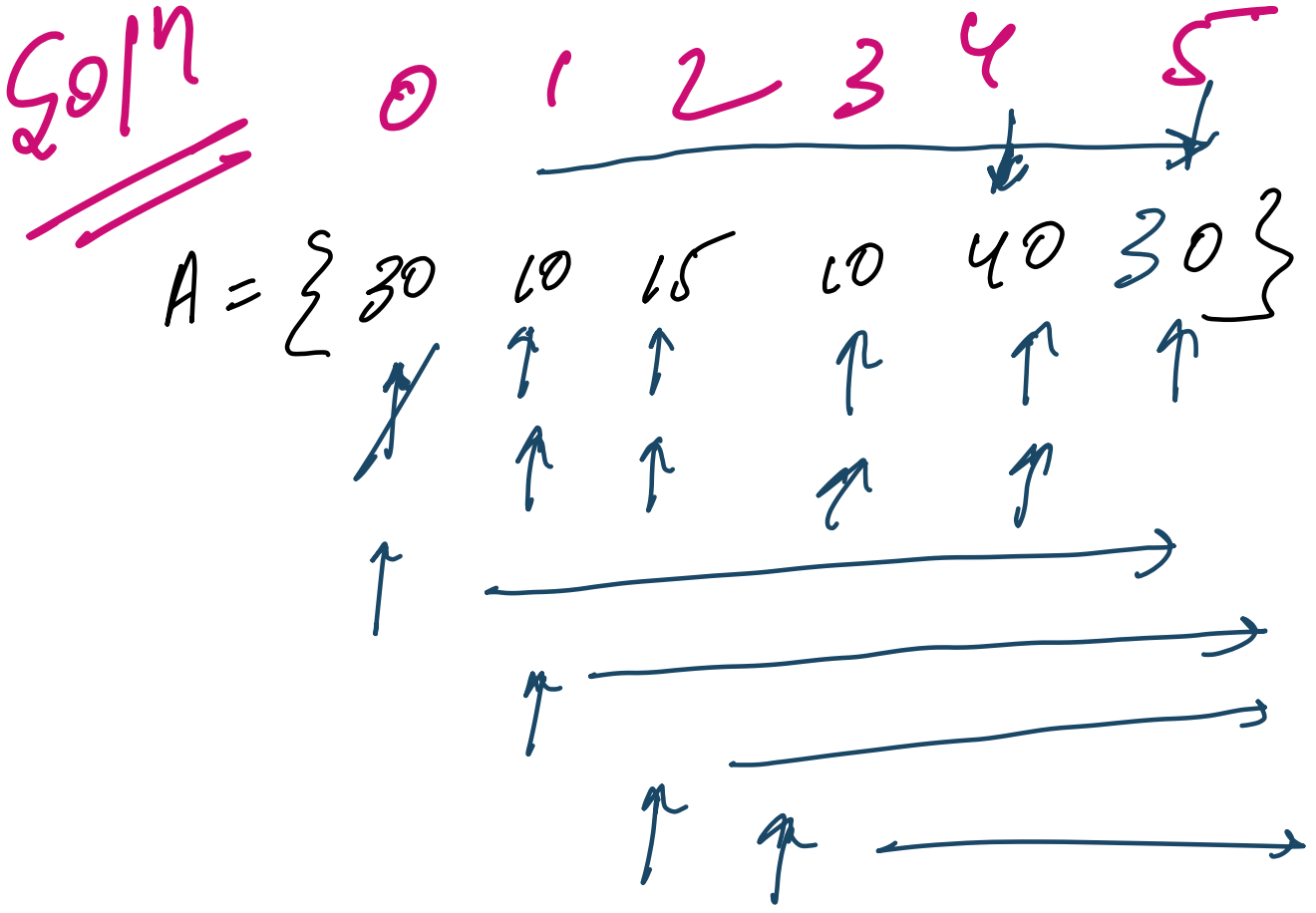
⇒ After Sorting



A = { 10 10 15 30 30 40 }

```
// Sorting logic
// (or)
// Arrays.sort(A);
for(int i=0; i<N-1; i++){
    if(A[i]==A[i+1]){
        S.O.println(A[i]);
    }
}
```

Soln



N = 6

```
for(i=0 ; i<N-1 ; i++){
    for(j=i+1 ; j<N ; j++){
        if(A[i] == A[j]
            print(A[i]);
    }
}
```

Q: odd even

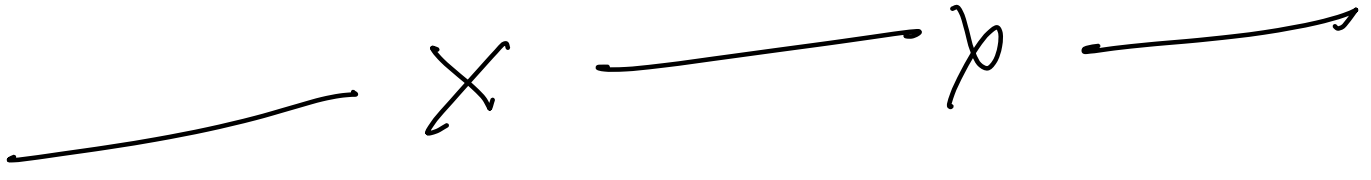
0 1 2 3 4 5 6

A = { 5, 3, 1, 2, 7, 8, 9 }

N%2 == 0 → Even

!=0 → odd

```
for( i=0 ; i<N ; i++) {
    if( A[i] %2 == 0) {
        S.O.P.ln( "Even" + A[i] );
    }
    else {
        S.O.P.ln( "odd " + A[i]);
    }
```



0 1 2 3 4 5 6

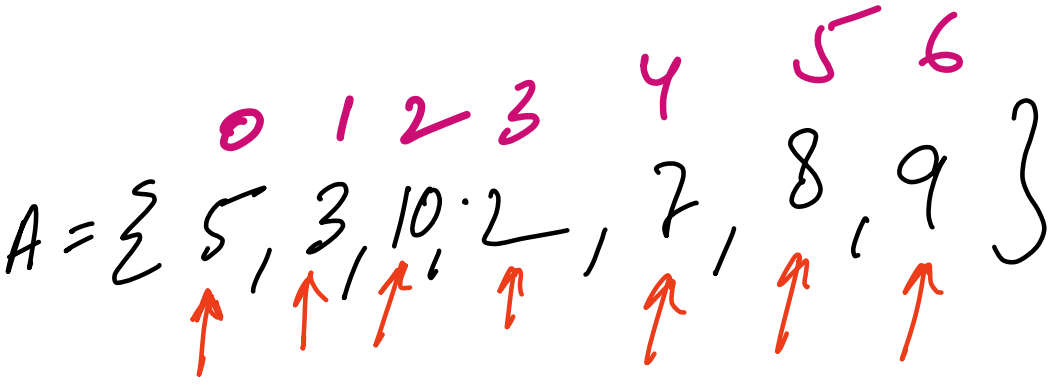
A = { 5, 3, 10, 2, 7, 8, 9 }

↳ N → length

```
int even_count = 0
int odd_count = 0
for(i=0; i<N; i++) {
    if(A[i] % 2 == 0) {
        even_count ++
    }
    else {
        odd_count ++;
    }
}
int[] even = new int[even_count]
int[] odd = new int[odd_count]
```

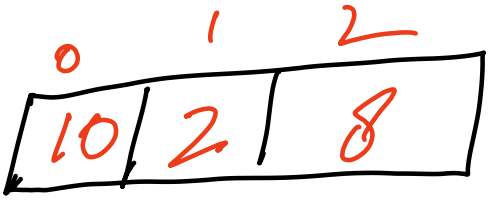
```
int even_index = 0;
int odd_index = 0;
for(i=0; i<N; i++) {
    if(A[i]%2==0) {
        even[even_index] = A[i];
        even_index++;
    }
    else {
        odd[odd_index] = A[i];
        odd_index++;
    }
}
```

Print odd & even Array

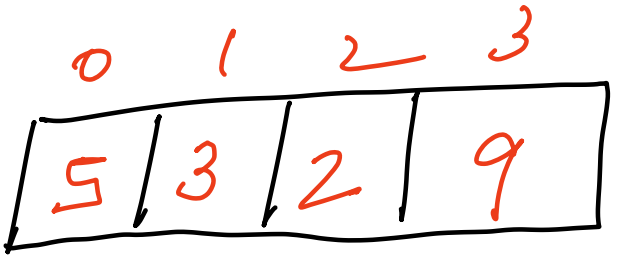


even_count = 3

odd_count = 4



even



odd

$i_e = 0 \not\ 1 \not\ 2 \not\ 3$

$i_o = 0 \not\ 1 \not\ 2 \not\ 3 \not\ 4$

Q: A = {10, 30, 50, 40, 20}

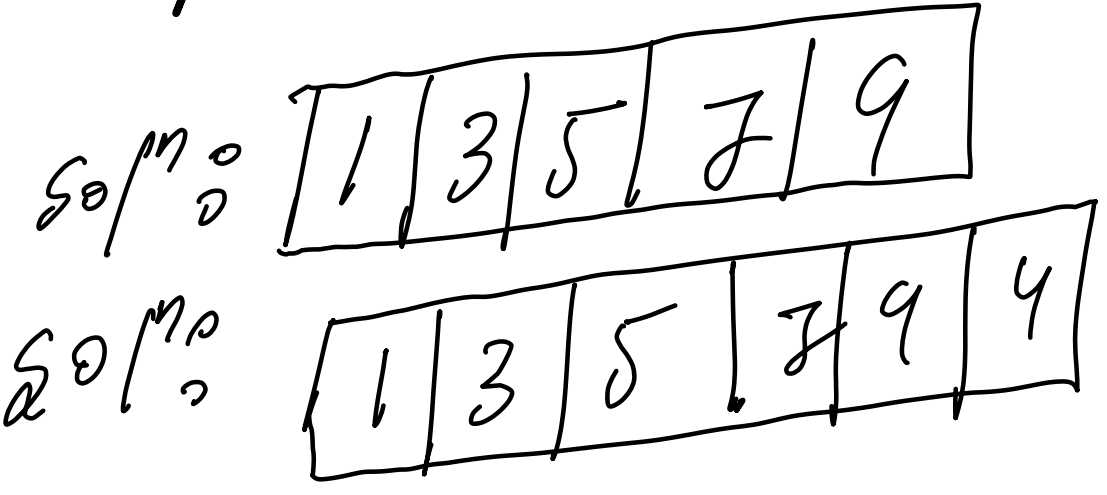
B = {70, 80, 10, 40, 90}

find duplicate elements in both Array

O/P ⇒ 10
40

Q: A = { 1 3 7 9 4 }
0 1 2 3 4

Insert 5 at 2nd index



* $A[i] = A[i-1]$;

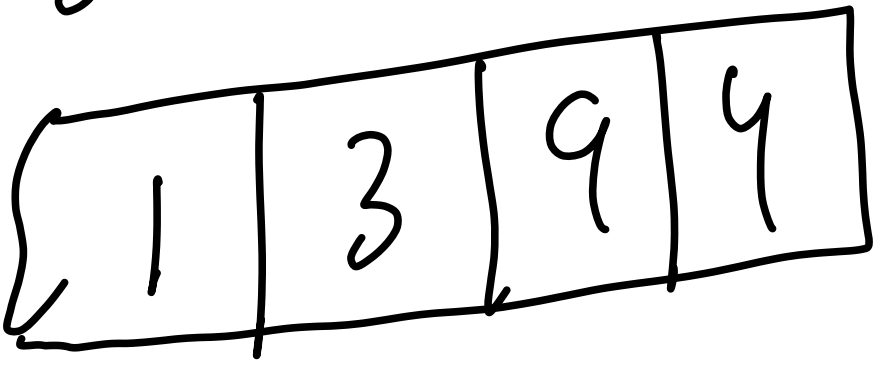
0 1 2 3 4

Q: A = { 1 3 7 9 4 }

delete position = 2

soln 1: { 1 3 9 4 4 }

soln:



*

Data Hiding

↳ Hiding data from outside user/class

↳ private access modifier

```
class Student {
    private int stdid ;
}
```

Soln: using public getter & setter.

Abstraction

↳ Hiding the details / implementation

Encapsulation → Data Hiding

Abstraction → Detail Hiding

Q: How to Achieve Abstraction ?

↳ Abstract class (0-100%)

↳ Interface → 100%

Abstract class

- Abstract method
- Concrete method

```
abstract class Test {
    //Concrete method
    void sum() {

    }
    // abstract method.
    abstract void show();

}
```

⟹ method overriding is used

⇒ If a class contains abstract method then class should declare abstract.

Q. Can a abstract class have Constructor → YES

Q. Can we create object of abstract class → NO

↳ But we can create Reference

~~Test t = new Test();~~

```
abstract class parent {
    abstract void show();
}

class child extends parent {

}
```

Parent p = new Child();

Parent p = ~~new~~ Parent();