1. **REMOVE DUPLICATES FROM SORTED ARRAY**

Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same. Then return the number of unique elements in nums.

Consider the number of unique elements of nums to be k, to get accepted, you need to do the following things:

- Change the array nums such that the first k elements of nums contain the unique elements in the order they were present in nums initially. The remaining elements of nums are not important as well as the size of
nums.
- Return k.

Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length
int k = removeDuplicates(nums); // Calls your implementation
assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be accepted.

Example 1:

Input: nums = [1,1,2]

Output: 2, nums = [1,2,_]

Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.

It does not matter what you leave beyond the returned k (hence they are underscores).

Example 2:

Input: nums = [0,0,1,1,1,2,2,3,3,4]

Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]

Explanation: Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.

It does not matter what you leave beyond the returned k (hence they are underscores).

Constraints:

1 <= nums.length <= 3 * 104

-100 <= nums[i] <= 100

nums is sorted in non-decreasing order.

2. **BEST TIME TO BUY AND SELL STOCK**

You are given an array prices where prices[i] is the price of a given stock on the ith day.You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Example 1:

Input: prices = [7,1,5,3,6,4]

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Example 2:

Input: prices = [7,6,4,3,1]

Output: 0

Explanation: In this case, no transactions are done and the max profit = 0.

Constraints:

1 <= prices.length <= 105

0 <= prices[i] <= 104

3. **EVALUATE REVERSE POLISH NOTATION**

You are given an array of strings tokens that represents an arithmetic expression in a Reverse Polish Notation.

Evaluate the expression. Return an integer that represents the value of the expression.

Note that:

The valid operators are '+', '-', '*', and '/'.

Each operand may be an integer or another expression.

The division between two integers always truncates toward zero.

There will not be any division by zero.

The input represents a valid arithmetic expression in a reverse polish notation.

The answer and all the intermediate calculations can be represented in a 32-bit integer.

Example 1:

Input: tokens = ["2","1","+","3","*"]

Output: 9

Explanation: ((2 + 1) * 3) = 9

Example 2:

Input: tokens = ["4","13","5","/","+"]

Output: 6

Explanation: (4 + (13 / 5)) = 6

Example 3:

Input: tokens = ["10","6","9","3","+","-11","*","/","*","17","+","5","+"]

Output: 22

Explanation: ((10 * (6 / ((9 + 3) * -11))) + 17) + 5

= ((10 * (6 / (12 * -11))) + 17) + 5

= ((10 * (6 / -132)) + 17) + 5

= ((10 * 0) + 17) + 5

= (0 + 17) + 5

= 17 + 5= 22

Constraints:

1 <= tokens.length <= 104

tokens[i] is either an operator: "+", "-", "*", or "/", or an integer in the range [-200, 200].

4. **SWAP 5 NUMBERS**

Say there are 5 variables as :-

a,b,c,d,e

You need to capture these values from user input.

Return the values after swaping these without using any extra variable as per beow condition:-

the value of a assign to e, value of b assign to d, value of c assign to a, value of d assign to c and value of e assign

to b.

Example :

Input:

a = 21

b = 25

c = 30

d = 40

e = 45

Output:

a = 30

b = 45

c = 40

d = 25

e = 21

Explanation: It should be swap with proper logic as can pass any of the integer value

5. **COUNT NUMBERS OF WORDS**

Say that any String is provided by User

You need to count the number of words in a string using Hashmap

Example :

Input:

Suppose the input String is : “This this is is done by Ashwin Ashwin”

Output:

{Ashwin: 2, by: 1, this: 1, This: 1, is: 2, done: 1}

Crud Operations

Details : Objective of application (CRUD) is to register user complaints and generate a ticket no.

Tech Stack : Spring, Hibernate, Mysql

Part 1 : Register Complaint (mandatory)

Create Fields :

Ticket no (auto generate,text), Date (auto current date and time, Time Stamp), Project Name (User Input, text), Module Name (User Input, text), Sub Module Name (User Input, text), Frequency (Always/Random), Priority (Critical/High/Medium), Explain Your Issue (Text), Status(New/Assigned/Resolved/Closed)

Note : All fields are mandatory

Column name - Date, Time, Ticket no, Project Name, Priority, Status,Module Name, Sub Module Name, Explain Your Issue, Frequency

Operations -

add - add complaint of users

Update - update complaint status

list - Show all complaints

getById - get Single complaint by id

getByTicketNo - get Single complaint by Ticket No

closeComplaint - user can close the status of complaint by id.

*manage history of complaint when complaint add and status changed.

* add loggers using log4J or slf4j

* handle exception like if id not present then throw “Data not found of id : <id>”