```
Given two string arrays word1 and word2, return true if the two arrays represent the
same string, and false otherwise.

A string is represented by an array if the array elements concatenated in order forms
the string.

Input: word1 = ["ab", "c"], word2 = ["a", "bc"]
Output: true

Input: word1 = ["a", "cb"], word2 = ["ab", "c"]
Output: false
```

[“ab”,”c”], [“a”,”bc”]

s1=abc s2=
bool areStringsEqual(vector<string>word1, vector<string>word2){
String s1=””, s2

=””;
for(int i=0;i<word1.size();i++){
s1+=word1[i];
}
for(int i=0;i<word2.size();i++){
s2+=word2[i];
}
if(s1==s2){
return true;
}
return false;
}

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]).

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49

[1,8,6,2,5,

area=width*height

Max area

6,5
height=min(leftHeight,RightHeight)
width=right-left

```
Int maxArea(int[] height){
Int left=0; //indexes
Int right=height.length-1;
Int maxArea=0;
while(left<right){
Int width=right-left;
Int h=Math.min(height[left],height[right]);
Int area=h*width;
maxArea=Math.max(area,maxArea);
if(height[left]<height[right]){
left++;
}
Else if(height[left]>height[right]){
Right–;
}
Else{
Left++;
Right–;
}
}
Return maxArea;
}
```

Time comp- O(N)
space-O(1)

1 2 3 4 5
arr=[1,1,1,2,3,3,4,4,5]
arr[i]!=arr[i-1]
c
Write a program to print unique numbers by removing duplicates from the array
[1,2,3,4,5]

nums[6]={0
}
Nums[1]=1

```
Void uniqueNumbers(int arr[]){
Int n=arr.length;
Int maxElement=arr[n-1];
Int nums[maxElement+1]={0}; //takes care of count
for(int i=0;i<n;i++)
{
        Nums[arr[i]]++;
}
//nums=[0,3,1,2,2,1]

for(int i=0;i<maxElement+1;i++){
if(nums[i]>0){
cout<<i<<" ";
}

}
```

1 2

[1,1,3,6,6,7]
1 3 6 7

nums=[0,2,0,1,0,0,2,1]
1 3 6 7

1 -> 2 -> 3 -> 4 -> 5 -> 6 -> null 1->2-null
2-> 1->4->3->6->5->null

```
ListNode* reverseInGroups(ListNode* head){
if(!head || head->next){
Return head;    1->2
2->null
}
prev=1, current=2, next=3 count=0
ListNode* Prev = null
ListNode* Current = head
ListNode* Next = null
Int count=0

while(current!=null && count<2){
Next=curent->next;
current->next=prev;
prev=current; //1
current=Next; 2
Count++; 1
}

if(Next!=NULL){
head->next=reverseInGroups(Next);
}

Return prev;

}
```

finding two numbers in an array that add up to a specific target sum. int[] nums = {2, 7, 11, 15}; int target = 9;
output={2,7}

i=0
j=i+1 1
{2, 7, 11, 15}
i=1
Mp:
2->0

Target-arr[i]

```
vector<int> twoSum(vector<int>& nums, int target){
unordered_map<int,int>mp;
for(int i=0;i<nums.size();i++){
if(mp.find(target-nums[i])==mp.end()){
mp[nums[i]]=i;
}
else{
return {nums[mp[target-nums[i]]], nums[i]}; //nums[mp[2]]=nums=[0]
}
}
return {-1,-1};
}
```

find the first non-repeating character and return its index. If it doesn't exist, return -1 String input = "ababc"
a->2 b->2 c->1
ans=c
i=4 index=-1 n=5
mpp[b]=1

"ababc"
mpp[c]

```
char firstUniqueCharacter(string s){
int index=-1;
int n=s.length();
map<char,int>mpp;
for(int i=0;i<n;i++){
mpp[s[i]]++;
}
i=0
for(int i=0;i<n;i++){
if(mpp[s[i]]>1){
continue;
}
else{
return s[i];
}
}
return ‘-1;;
}
```

1. getEmployee
2. getAllEmployees
3. updateEmployee
4. deleteEmployee

```
Class EmployeeController{
@Autowired
Private EmployeeService empService;

@GetMapping
Public List<Employee> getAllEmployees(){
Return empService.getAllEmployees();
}
```

```
@GetMapping(“/{id}”)
Public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id){
Return empService.getEmployeeById(id).map(employee -> new
ResponseEntity<>(employee,HttpStatus.OK))
}
```

Abs

```
Abstract Class Shape{
Public abstract void draw();
Public void commMethod(){
System.out.println(“Hello i am abstract”);
}
}
```

Interface

```
Interface Drawable{
Void  draw();
Default void commMethod(){
System.out.println(“Hello i am interface”);
}
}
```

Order fk as customerId
Customer pk customerId

```
Select Order.customerId As order_id, Customer.customerId AS customer_id
FROM Order
INNER JOIN Customer ON Order.customer_id=Customer.customer_id;
```

String input = "Hello World" output = "dlrow olleh"

```
Void reverseString(string input){

Int start=0,end=input.length()-1;

while(start<end){

Char temp = input[start];

input[start=input[end];
```

```
input[end]=temp;

Start++;

End- -;;

}

}
```