Array Problems - I

⇒ **Reverse An Array -**

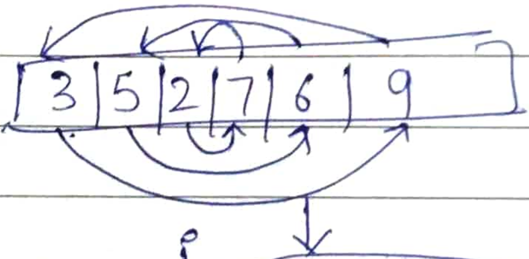
I/P

3	5	2	7	6	9
0	1	2	3	4	5

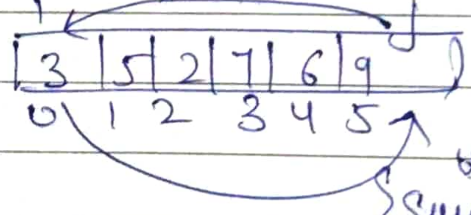
O/P

9	6	7	2	5	3
0	1	2	3	4	5

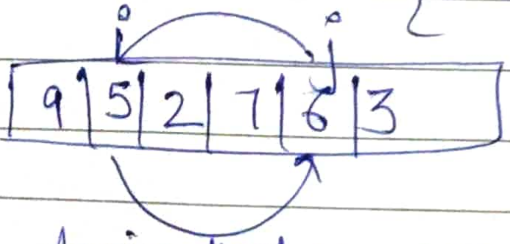
Approach - Ulta for loop lga ke print krdo → this is wrong because we don't actually reverse the array, only print in reverse.



Correct Approach



```
i=0
j=n-1
while(i<j)
{swap(arr[i], arr[j])
i++;
j--; }
```



Again keep on swapping till (i<j)

Code of ~~swap~~ Reverse →

```
int i=0
int j=n-1
while(i<j)
{
swap(arr[i],arr[j])
 i++;
 j--;
}.
```

H/W → XOR se swap kaise krte h?
→ + ya – ka use krke swapping.

⇒ Max. element in array –

3	12	7	18	17	16

↑

maxi.

I)
```
int maxi =INT_MIN;
for( i=0; i<n; i++ )
 {
 ans=max(ans, arr[i]);
 }
return ans;
```

II)
```
int  ans=arr[0]
for( i=0; i<n; i++)
 {
  ans=max(ans, arr[i]);
    }
return ans;
```

```
    getmax(a,b)
III) int getmax (int a,b)
     {
      if(a>b)
        return a;
      else
      return b;
     }
```

H/W - Find min. element in an array.

Max ya fir ans me INT_MINse hi kyu initialize krte h?

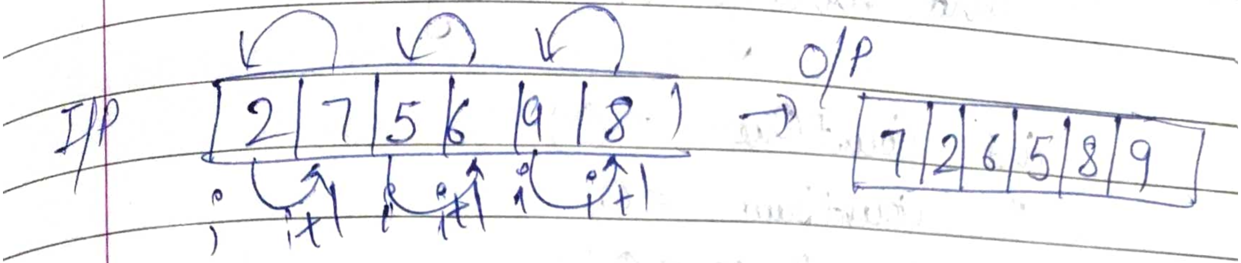
maxi= INT_MIN →
ans = INT_MIN → } why?

lets say array has - | -1 | -2 | -3 |
$\boxed{ans = -1}$

If we would have initialized ans with 0, then answer -1 ki jagah 0 aa jata jo ki galat hota.

Thats why to prevent this case, we use INT_MIN except for any other number.

⇒ Swap alternates in an array –



Observations –

→ i, i+1 ko swap krna h.

→ i = i+2 → i ko 2 se aage badha rhe h.

```
while(i<n){
if(i+1<n)
 swap(arr[i],arr[i+1]
i=i+2 }
```

why? because if we check (i<n) inside loop and then try to swap arr[i] & arr[i+1], then in case (i+1) is not in array, it will be out of index and "code phat jaega."

OR

```
while(i+1<n)
{
swap(arr[i],arr[i+1]);
i=i+2;
 }
```

⇒ Sort an array of 0's, 1's, 2's.

I/P → 3,5,2,1,7
Sort in incr/dec. order.
O/P → 1,2,3,5,7

Only 0's, 1's & 2's –
I/P → 1 0 2 2 0 1 1
O/P → 0 0 1 1 1 2 2

Approach - Count the no. of 0's, 1's & 2's and arrange in array.

```
int  CountOne   =0;
 "   CountTwo   =0;
 "   CountThree =0;

for ( int i →0→<n )
  {
   if (arr [i]==0)
   countOne ++;
   else if (arr[i] == 1)
   countTwo ++;
   else
   countThree ++;
  }
for ( int i→0<n )
 {
// put 0's, 1's & 2's in array as
per their count.
  }.
```

→ **Sort** : predefined function -

```
sort (arr, arr + size);
```

↓

ye array ko sort kardega.

<u>full Code</u> -

```
void SortOneTwoZero( int arr[],int n)
{
 int one=0, zero =0, two =0;
for (int i=0; i<n; i++)
 {
 if (arr[i] ==0)
  zero++;
 else if (arr[i] ==1)
  one++;
 else
  two++;
 }
int i=0;
//put zero
while (zero--)
 {
 arr[i] =0;
  i++;
 }
 //put one
 while (one--)
 {
 arr[i] =1;
  i++;
  }
  // put two
  while(two--)
  {
   arr[i]=2;
   i++;
  }  }
```

```
short form of -
 while (zero!=0)
  { arr[i] =0;
    i++;
    zero--;
   }.
```

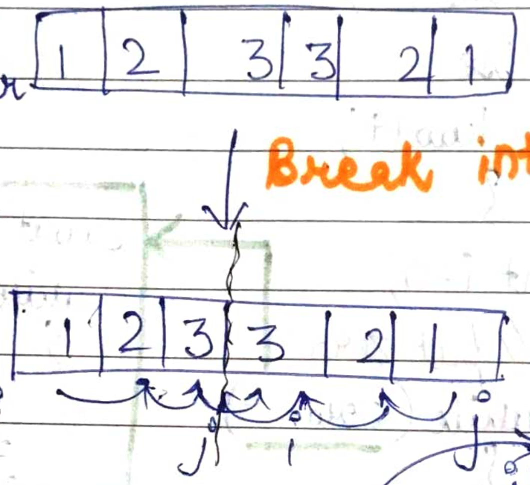
ye loop tab tak chalega jab tak mera zero wala variable 0 ke equal na ho jaye.

Palindrome → same when read from start or from end.

eg. $\overrightarrow{\text{MADAM}}$ ←

or 12 33 21 ↓reverse 123321 } equal = palindrome

Approach - arr



i=0
j=n-1
if(arr[i]==arr[j])
{i++;
j--;}

while(i<j)

because it should stop when (i>j).

Code - ~~vo~~

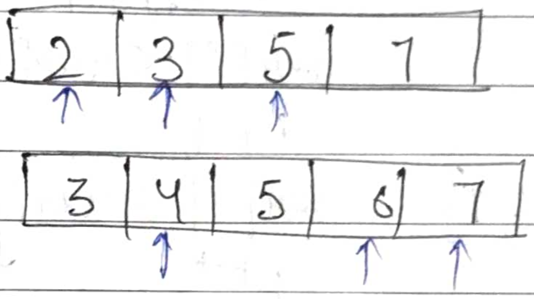
```
bool palin (int arr[], int n)
{
  while (i < j)
  {
  if (arr[i] == arr[j])
   {
     i++;
     j--;
   }
else
return false;
}
return true;
}.
```

Union & Intersection of Two Sorted Arrays -

Union -

I/P → arr1

arr2

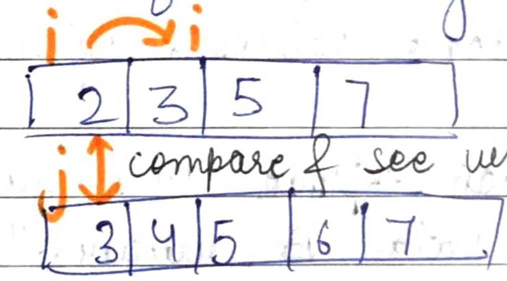


O/P → Union → { 2,3,4,5,6,7 }

Approach - Take two pointers i & j from start of both arrays.

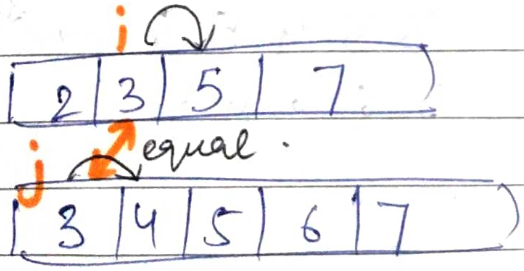
arr1

arr2

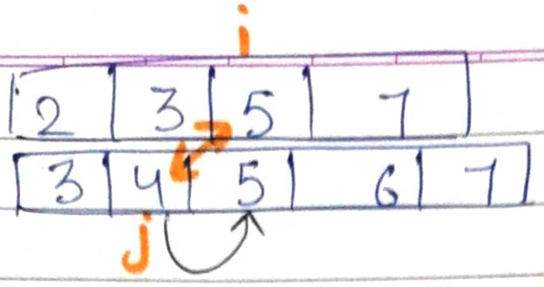


compare & see which one is less.

a[i] is smaller, so put it in answer array & i++.



arr[i] = = arr [j]
then i++ & j++ and put it in answer.



arr[i] > arr[j]
then put ~~&~~ arr[j] in answer and increment j. {j++}

Three cases - arr[i] == arr[j]
i++, j++
arr[i] < arr[j]
i++
arr[i] > arr[j]
j++

ans[] =
{2,3,4,5,6,7}

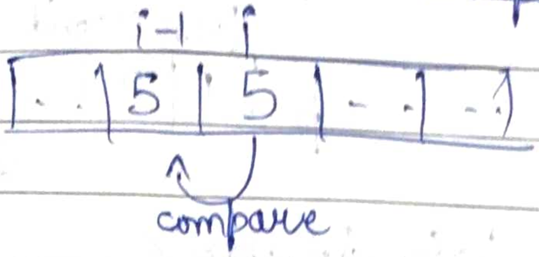
Fault in Approach-

If any array has duplicate values, then answer will also contain duplicate values, which is wrong.

* You can explore this question using "Set Data Structure" which does not store duplicate values.

eg
1 1 2 2 2
3 3 4 5 5 → size of set
5 6 6 6 6   ↳{6}.

Modified Approach - Check (i-1) value when you check i^th value and compare if its equal.



If i & i-1 values are same then aage badh jao, without adding it to answer.

H/W → Khud ye code krne ka try kro.

Intersection - same values.
Reuse the condition → arr[i] == arr[j].
and store into answer.

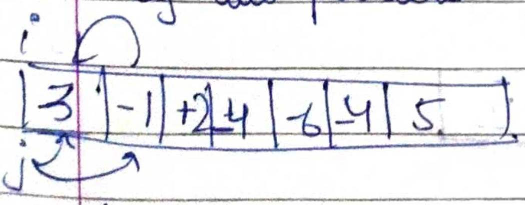
Move -ve No.'s to one side of array.

I/P → [3 | -1 | 2 | -4 | -6 | -4 | 5]

O/P → [-1 | -4 | -6 | -4 | 3 | 2 | 5]

Approach - Apply loop and compare by two pointers.

```
for (int i=0 → <n)
{
if (arr[i] < 0)
{
swap (arr[i], arr[j])
    j++;}
}
}
```



Pair – pair <int, int>
p = make_pair (1,2);

first second
p $\boxed{1 | 2}$

→ creates a block in memory which contains 2 values of any datatype.

pair <int, char>
pair <char, char>

Find duplicate in an array of N+1 integers.

arr[] = { 1, 2, 3, 3, 4 }

Cond^n:
↳ values are from 1 → N but 1 No. is repeating. You need to find it.

{ 1, 2, 3, ... x . . . N }
↑ duplicate.

arr[] = { ~~1~~, ~~2~~, ~~3~~, 3, ~~4~~ } → answer.
arr2[] = { ~~1~~, ~~2~~, ~~3~~, ~~4~~ }

① Input array ka sum nikal lo.
② Also find another sum of (1, 2, 3 . . . N) by formula

$$\frac{n*(n+1)}{2}$$

$$\boxed{ans = sum1 - sum2}$$

```
int sum1 =0;
for(int i=0 →<n)
  {
   sum1 =sum1 + arr[i];
    }
 int sum 2 = n*(n+1)/2;
 int ans = sum1 - sum2;
  return ans;
```

check for a condition → if (sum > range)

INT_MIN ka
use kro.

Kadane's Algo – Largest Sum Contiguous Subarray.

-2	-3	4	-1	-2	1	5	-3

→7

```
int getMaxSubArraySum(int arr[], int n)
{
  int maxSF  =INT_MIN;
  int maxEH =0;
for(int i=0; i< n;i++)
    [
  maxEH = max EH +arr[i];
 max SF = max (maxSF ,max EH);
 if (max EH <0)
     maxEH =0;
 }
 return maxSF;
      }
```