EASY

ARRAY Q/S

Q1 Remove~~s~~ Duplicates from sorted Array.

Approach ~~1st~~

1. Take a variable i as 0;
2. Use a for loop by using a variable 'j' from 1 to length of the array
3. If arr[j] != arr[i], ↑ 'i' and. update arr[i] == arr[j].
4. After completion of loop. return i+1, i.e size of array of unique elements.

Date.................. Day..................

Code

```
int i = 0;
for (int j=1; j<n; j++){
    if(arr[i] != arr[j]){
        i++;
        arr[i] = arr[j];

    }
}
return i+1;
```

Parent's Sign. .... Teacher Sign. ....

Q2 Largest Element in Array.

Code

```
int maxValue = arr[0];
for (int i=0; i<n; i++){
    if (arr[i]>maxValue){
        maxValue = arr[i];
    }
}
return maxValue;
```

Q3 Return Second largest element.

Code

```
int print2largest(int arr[], int n){
    sort(arr, arr+n);
    int second=0;
    for(int i=n-2; i>=0; i--){
        if(arr[i]!=arr[n-1]){
            second = arr[i];
            return second;
        }
    }
    return -1
}
```

N=6

{12,35,1,10,34,1}

Starts iterating from here.

Q4 Check if array is sorted.

Code

```
bool arraySortedORNot (int arr[], int n){
    //code here.
    if(n==1){
        return true;
    }
    int c=0;
    int max= arr[0];
    for(int i=1; i<n; i++){
       [if (arr[i]>=arr[i-1] && arr[i]>=max){
            max= arr[i];
            c++;
        }
    }
    if (c==n-1){
        return true;
    }
    else{
        return false;
    }
```

Basically a count variable.

{ i,j,i,.. } if greater. then not sorted.

Date.................. Day.................

Q5 Quick Left Rotation.

Approach ~~rotate~~ Let Array be broken in A & B where A = arr[0 . . . d-1] & B = arr[~~0~~d . . . n-1]

① Reverse A to get A(r)B
↳ reverse of A.

② " B to get A(r)B(r)

③ " all to get BA.

Code

```
void reverse (int arr[], int i, int j){
        while (i<j) {
              swap(arr[i++], arr[j--]);
        }
}
```

Date.................. Day.................

```
public:
void left Rotate(int arr[], int K, int n).
{
28 ←        int d = K%n;
            if (d=0){
                return;
            }
            reverse(arr, 0, n-1)  //all
            reverse(arr, 0, n-d-1) // A(r)
            reverse(arr, n-d, n-1) // B(r)
}
```

Date............... Day...............

Q6 Move all zeroes to end of Array.

Approach: ① Start traversing from the first occurrence index of zero.

② Take 2 variables (i, j) - "i" will be first occcurance of zero and "j" is i+1

③ if element at j index is not zero then swapp elements at i, j and the increment i, j

④ if the element at j index is zero, then only increment j.

Day...............

Date...............

code

```
// finding first zero occurance.
    int k=0;
    while(k<n){
        if (arr[k]==0){
            break;}
        else{
            k=k+1;}

    }
    // finding zeroes & immediate non
       zero elements & swapping them.
    int i=k, j=k+1;
    while(i<n && j<n){
```

Parent's Sign. .... Teacher Sign. ....

```
if (arr[j]!= 0) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
    i++}
}   }j++;
```

~~Imp~~

Q7 Find Union of two Unsorted Arrays.

Approach Using Set

Eg arr1[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

arr2[] = {2, 3, 4, 4, 5, 11, 12}

arr = arr1 + arr2 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 3, 4, 4, 5, 11, 12}

Union = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

Code

```
vector <int> findUnion (int arr1[], int arr2[], int n, int m)
    set <int> s;
    vector <int> Union;
    for(int i=0; i<n; i++)~~}~~ {
        s. insert(arr1[i]);
    for(int i=0; i<n; i++){
        s. insert (arr2[i]);
                                          ~~vector <int~~
    for(auto& it : s){
        Union. push-back(it);
    return Union;
```

Used as.
for(int i=0, i<n, i++)

Q8 Missing Number. (Amazon)

Approach Eg A[] = {0, 1, 2, 3, 5}

[1 to 5] ~~Sum of Array~~ Total Sum. $= \frac{n(n+1)}{2} = 15$

Sum of Array = 0+1+2+3+5 = 11

* Total Sum = ReqNum + Sum of Arr.

Code

```
int SumArr = 0;
for (int i=0; i<N-1; i++) {
              or i<n
                 num.size()
        sumArr += A[i]
}
int ExpSum = (n*(n+1))/2;
int ReqNum = ExpSum - SumArr;
return ReqNum;
```

Q9 Maximum number of consecutive 1's.

Approach Eg arr[] = { 0, 1, 1, 1, 0, 0, 1, 1, 1, 1 }
0 1 2 3 4 5 6 7 8 9

① Take "cnt" point ~~a~~ iterate it thru array.

② When nums[i] == 1, cnt++

③ Initialize "maxi" variable with 0

④ max(maxi, cnt) = maxi to find maximum number of consecutive 1's.

code

Date.................. Day..................

```
int cnt = 0;
int maxi = 0;
for(int i = 0; i<nums.size(); i++){
    if (nums[i] == 1){
        cnt++;
    }
    else {
        cnt = 0;
    }
    maxi = max(maxi, cnt);
}
return maxi;
```

Q10 Reverse an Array (2 pointer approach)

Eg

0 1 2 3 4

11, 7, 3, 12, 4

s e

① swap s&e → 4, 7, 3, 12, 11 (s, e)

② " " " → 4, 12, 3, 7, 11 (s e)

③ " " " → $\boxed{4, 12, 3, 7, 11}$ Ans

Code

```
int s = 0;
int e = v.size()-1
while( s<=e) {
        swap.(v[s], v[e]);
        s++;
        e--;
}
return v;
```

Q11 Merge sorted Array.

Approach

arr1[] = {1, 3, 5, 7, 9} (i)

arr2[] = {2, 4, 6} (j)

if i<j,

put i first then j

base: while(i<n && j<n)

Date.................

Day.................

Code

```
int i=0, j=0;k=0;
while (i<n && j<m) {
    if (arr1[i]<arr2[j]) {
        if (arr1[i] ~~≤~~<arr2[k]){
            arr3[k]= arr1[i];
            k++;
            i++;}
        else{
            arr3[k]= arr2[j];
            k++;
            j++;
```

Date................

```
// copy first array k element ko
while(i<n){
    arr3[k] = arr1[i];
    k++;
    i++;
}
// copy Kido second array k remaining element
ko
while (j<m){
    arr3[k] = arr2[j];
    k++;
    j++;
}
```

Teacher Sign. ....   Parent's Sign. ....

Make each day your masterpiece.