L-7 Single Dimensional Arrays.

Array =

- A Single Variable, which reference large collection of data is called array.

We not need to assign variable again and again.

- Declaring Array.

⇒ DataType[] Arrayname.

eg ⇒ double[] mylist;

- Creating an array.

⇒ arrayname = new datatype[value]

- Declared and Created in One Statement.

⇒ datatype[] arrayname = new datatype[size]

eg-

double[] mylist = new double[10];

⊙ Array Size and default Values

⇒ To want an arraySize.

arrayReVar.length.

myList.length = 10.

⊙ Array Initializers.

elementType[] arrayName = {value0, value1, ..., valuek};

eg:-

double[] myList = {1.9, 2.9, 3.4, 3.5};

main inside = 10^6.
Global array = 10^7. } maximum size of array.

Processing Arrays.

① Make array user input Values.

```
java.util.Scanner input = new java.util.Scanner
                          (System.in);
for (int i = 0; i < mylist.length; i++)
    mylist[i] = input.nextDouble();
```

② Array with random Values.

```
for(int i = 0; i < mylist.length; i++) {
    mylist[i] = Math.random() * 100;
}
```

③ To display an array

```
for(int i = 0; i < mylist.length; i++){
    System.out.println(mylist[i] + " ");
}
```

* Arrays *

This is Smart Notes, Detail Explain in different notes.

Easy - 14.

① Largest element in array.
{2, 5, 1, 3, 0} ⇒ find max.

⇒
```
int max = arr[0];
if (arr[i] > max)
    max = arr[i]
}
return max.
```

Ek max naam ka variable lege.
agar koi bhi usse bada hoga to.
arr[i] ki value max ko dedege.
Phir max ko return kardege.

② Second Largest and Second Smallest.

⇒ two Small Value = Large, SecLarge.

if any Element.
Greater than.
Large > arr[i].

SecLar = Large, Large = arr[i]

(arr[i] > SecLar && arr[i] != Large)
SecLar = arr[i]
}

Same as Second Small.
two big value = Small, SecSmall

③ Check if array is Sorted.

⇒ we Compared both Values.
if agar pehli value badi hogyi aage se to.
Sorted Nahi hai.
hamesa aage ki value badi honi chaiye.

```
for (int i = 1; i < n; i++){
    if (arr[i] < arr[i-1]){
        return false
    }
    return true;
}
```

④ Remove Duplicated from Sorted array.

Eg - {1, 1, 2, 2, 2, 3, 3}.
ans ⇒ {1, 2, 3}.
return = 3.

i and j Compare.
if not Equal.
then
(j ki value i) ki Jaga pe.
jayegi aur i++, j++ Last me return i+1;

arr = [1, 1, 2, 2, 2, 3, 3]
i j

arr[i] != arr[j] i++
arr[i] = arr[j]

Code.

```
int i = 0;
for (int j = 1; j < arr.length; j++){
    if (arr[i] != arr[j]){
        i++;
        arr[i] = arr[j];
    }
} return i+1;
```

⑤ Left Rotate by One place.

Eg - 1, 2, 3, 4, 5.

Ans ⇒ 2, 3, 4, 5, 1

temp = arr[0]

arr[i] = arr[i+1]

= arr[n-1] = temp

```
int temp = arr[0];
for
```

⑥ Left Rotate by k element.

Eg ⇒ {1, 2, 3, 4, 5, 8, 9}.

k = 3.

{4, 5, 8, 9, 1, 2, 3}.

⇒

is m, phale half ko reverse. karege jitna k hota hai phir remaing part, aur last m whole part.

{1, 2, 3, 4, 5, 8, 9}. k = 2

reverse 3, 2, 1

9, 8, 5, 4

reverse whole.

= ans 4, 5, 8, 9, 1, 2, 3.

Reverse Code.

```
while (start <= end){
int temp = arr[start];
arr[start] = arr[end];
arr[end] = temp;
start++;
end--;
}
```

Page No.
Date

(7) Move all Zeros End of the array.

Eg ⇒ {1,0,2,3,0,4,0,1}

{1,2,3,4,0,0,0}

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

    int temp = a[i];
    a[i] = a[j];
    a[j] = tmp;
      j++;
  }
}
```

(8) Linear Search.

```
for(int i=0; i<n; i++){
if(arr[i] == num){
    return i;
}
return -1;
```

(9) Union of two Sorted Arrays

arr1[] = {1,2,3,4,5}
arr2[] = {2,3,4,5}
Output = {1,2,3,4,5}

```
int i=0, j=0;
ArrayList<Integer> Union = new ArrayList();
while(i<n && j<m){
  if(arr1[i] <= arr2[j]){

if(Union.size()==0 || Union.get(
  Union.size()-1) != arr[i])
  Union.add(arr[i]);
    i++;
} else {

if(Union.size()==0 || Union.get
  (Union.size()-1) != arr2[j])
    Union.add(arr2[j]);
      j++;
}
while(i<n){
  if(Union.get(Union.size()-1) != arr
    Union.add(arr1[i]);
      i++;
}
while(j>m){

    j++;
}
```

Page No.
Date

(10). Find missing number in an array.

N=5, arr[] = {1, 2, 4, 5}
Result = 3.

xor1 = {1^2^3^4^5).

xor2 of arr = {1^2^4^5).

xor1 ^ xor2
= (1^2^3^4^5) ^ (1^2^4^5)
= (1^1)^(2^2)^(3)^(4^4)^(5^5)
(0)^(0)^(3)^(0)^(0)
= 3

```
int xor = 0, xor2 = 0;

for(int i=0; i<N-1; i++){
    xor2 = xor2 ^ a[i];
    xor1 = xor1 ^ (i+1);
}
xor1 = xor1 ^ N;
return (xor1 ^ xor2);
```

(11). Maximum Consecutive One's in the array.

prices = {1, 1, 0, 1, 1, 1}
Output = 3
2's = 1, 3's = 3.

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

(13) Longest Subarray with Given Sum k

```
=> (int [] a , long k) {

int n = a.length;

int left = 0, right = 0;
long sum = a[0];
int maxlen = 0;
while (right < n){

while ( left <= right && sum > k){
sum -= a [left];
left ++;
}

if (sum == k
maxlen = Math.max (maxlen,
right - left + 1);
}

right ++;
if ( right < n) sum += a[right];
}

return maxlen;
}
```

left aur right = a[0]. (starting) add right ++ agar wo target k se jayda to left ko --; aur ek maxlength le lena tabi wo check karega. Sab barabar hai ki nahi.

(14)

```
(int [] a , int k){

int n = a.length;

int sum = 0;
int maxlength = 0;

for(int i=0; i <n; i++){

sum += a[i];

if (sum == k){
maxlen = Math.max (maxlen, i+1);
}
int rem = sum-k;

if (preSumMap.contains(rem)){

if (!preSumMap.contains(sum)){
preSum.put (sum, i);
}
}

return maxlen;
```

* Medium Problem of Arrays *

Page No.
Date

2Sum Problem.

(1) arr[] = {2, 6, 5, 8, 11}.
target = 14.

⇒ koi bhi 2 number ko add. karne par target milna. chaiye.

① Bruteforce (i → 0 to n, j → i to n, == target)
② Better = Hashing.
③ Best. → ① Sort
② left + Right
if Greater: Right--;
if Smaller: left--;

Code.

```
Arrays.sort (arr);
left = 0;   Right = n-1;
While ( left < right) {
int Sum = arr[left] + arr[right]
  if (Sum == target)
      return "YES"
  } if (Sum < target) left++;
    else right--;
} return NO.
```

(2) Sort an array 0's, 1's and 2's.

⇒ nums = [2,0,2,1,1,0].
Output = [0,0,1,1,2,2].

3 Rules
if(0) low ↔ mid, mid++ low++
if(1) mid++
if(2) = mid ↔ high, high--;

[2, 0, 2, 1, 1, 0]
m ↑ ↑ h

low → [2, 0, 2, 1, 1, 0] ← high
mid →

```
While (mid <= high) {
  if (arr.get (mid) == 0) {
    int temp = arr.get (low);
    arr.Set( low, arr.get(mid));
    arr.Set (mid, temp);
    low++ , mid++;
  } else if (arr.get (mid) == 1) {
      mid++;
  } else {
    int temp = arr.get (mid);
    arr.Set( mid, arr.get (high));
    arr.Set( high, temp);
      high--;
  }
}
```

(3) Majority Element n/2 times.

```
int n = V.length;
int cnt = 0;
int el = 0;
for(int i = 0; i<n; i++) {
  if(cnt == 0) {
    cnt = 1;
    el = V[i]
  } else if (el == V[i]) cnt++
    else cnt--;
  }
  if (cnt1 > (n/2)) return el
    return -1;
```

④. Kadane's Algorithm.
maximum Subarray.

```
for(int i= 0; i<n; i++){
  Sum  t= arr[i];

  if( Sum > maxi ){
    maxi = Sum;
  }
  if (Sum<0){
     Sum=0;
  }
} return maxi.
```

above Extended.

⑤ Print Subarray with.
maximum Subarray

```
long maxi = Long.MIN.Value;
long Sum = 0;

int Start=0;
int anoStart = -1; anoEnd=-1;

for(int i=0; i<n; i++){.

if(Sum ==0) Start = i;

Sum  t= arr[i]
if (Sum > maxi ){
  maxi = sum;
  anoStart = start;
  anoEnd = i;
} return;
```

⑥. Stock Buy and Sell.

prices = [7, B 1, 5, 3, S 6, 4]
output = 5.
6 - 1 = 5.

Code

```
int maxPro = 0;
int minPrice = Integer.MAX_Value;

for( int i=0; i< arr.length; i++){

minPrice = Math.min(minPrice, arr[i]
maxPro = Math.max (maxPro,
          arr[i] - minPrice);
}
return maxPro;
```

⑦. Rearrange Element by Sign.
arr[] = {1, 2, -4, -5} N=4.
output = 1, -4, 2, -5.

⇒. Pos ⇒. Arraylist
neg

```
Arraylist<Integer> Pos = new Arraylist<>()
neg.
phir arrange
      Even = Pos   2*i
      odd at = neg  2*i+1
```

⑨. Leaders in Arrays.

{ 10, 22, 12, 3, 0, 6 }

Set {

{ 6, 12, 22, max =

int max = arr[n-1];

ans.add (arr[n-1]);

for(int i=n-2; i>=0; i--){

if (arr[i] > max){

ans.add (arr[i]);

max = arr[i];

}

return ans;

⑧ Next Permutation.

Arr[] = {1, 3, 2}

Output = {2, 1, 3}.

Step-1 ⇒. find BreakPoint means start backward and find which is Greater than Greater.

2 1 5 4 3 0 0

1

Step-2 ⇒. 2 1. but Smallest among them and Swap.

Step-3 ⇒. Reverse.

int ind=-1;

for(int i=n-2; i>=0; i--){

if(A.get(i) < A.get(i+1)){

ind = i;

break;

}

}

⑩. Longest Consecutive Sequence in an Array.

for(int it : Set){

if(!Set.contains(it-1){

int cnt = 1;

int x = it;

while(Set.contain(x+1)){

x = x+1;

cnt = cnt+1;

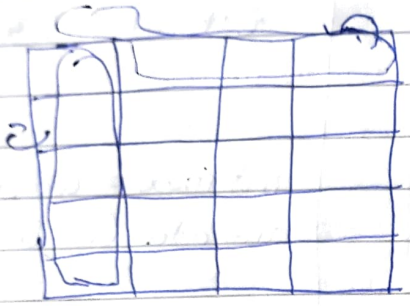
}

longest = Math.max(longest, cnt);

}

return longest;

(11). Set Matrix Zero.



(12). Rotate by 90 degree.

(1). Transpose the Matrix
(2). Reverse Each row of matrix.

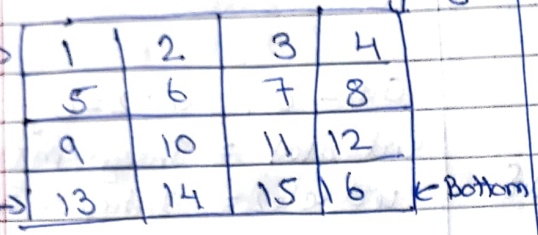
```
for(int i = 0; i< matrix.length; i++){
 for(int j = i; j < matrix[0].length; j++){

  int temp = 0;
   temp = matrix[i][j];
matrix[i][j] = matrix[j][i]
 matrix[j][i] = temp;

for(int i=0; i< matrix.length; i++){
 for(int j=0; j < m/2 ; j++){
  int temp = 0;
   temp = matrix[i][j];
matrix[i][j] = matrix[i]
matrix[i][m.l. -1-j] = temp;
```

Page No.
Date

Spiral Traversal.



Code

```
top=0; left=0, b= n-1; right=m-1;

while (top<= bott && left <=right){.

for(int i= left; i<=right; i++)
    ans.add (mat [top][i]);
     top++;

for (int i= top; i<= bottom; i++).
    ans.add (mat[i] [right]];
     right --;

 if (top<= bottom){.
for (int i= right; i>=left; i--)
    ans.add (mat[bottom][i]);
     bottom--;

if (left<= right){
   for (int i= bott; i>=top; i--)
    ans.add (mat[i][left]);
    left++;
  }
}

return ans;.
```

Count Subarray with Given Sum.

```
int n= arr.length;
Map mpp= new HashMap ();
int preSum=0, Cnt=0;

mpp.put (0,1);

for (int i=0; i<n; i++){.
  preSum += arr[i];

int remove = presum-K;

Cnt += mpp. (remove, 0);

mpp.put (presum, mpp+1);

return Cnt;.
```

Page No.
Date

Hard.

① Pascal's Triangle.

```
for (int col=1; col < row; col++){
    ans = ans * (row - col);
    ans = ans / col;
ansRow.add((int) ans);
}
return ansRow;
```

② Majority Element (N/3)

```
cnt1 = 0, cnt2 = 0;
int ele1 = Integer.MIN
int ele2 = Integer.MIN;

for(int i=0; i<n; i++){
 if (cnt1==0 && ele2 != v[i]){
     cnt1 = 1;
     ele1 = v[i];
 } else if (cnt2==0 && ele1 != v[i]){
     cnt2 = 1;
     ele2 = v[i];
 } else if (v[i] == ele1) cnt1++;
 else if (v[i] == ele2) cnt2++;
  else{
     cnt1--; cnt2--;
  }
}
```

Page No.
Date

③ 3-Sum Problem.

i	j				k
1	2	3	4	5	6

```
while (j<k){
int sum = a[i]+a[j]+a[k];
    if (sum<0)
      j++;
  } else if (sum>0){
      k--;
    } else {
    ans.add(temp);
      j++;
      k--;
while (j<k && arr[j]==arr[j-1])j++;

while (j<k && arr[k]==a[k+1]) k--;
}
return ans;
```

④ 4-Sum Problem.

i	j		k				l
10	20	30	40	50	60	90	20

Page No
Date

(5). Longest Subarray.

```
int maxi = 0;
int Sum = 0;

for(int i = 0; i < n; i++){

  Sum += A[i];

  if (Sum == 0){
    maxi = i + 1;
  }
  else {
  if (mpp.get(Sum) != null) {

maxi = Math.max (maxi, i - mpp.get(Sum));
  }
  else {
      mpp.put (sum, i);
  }
  return maxi;
```

(6). Count number of Subarrays with Given Xor k.

```
Map< Integer, Integer> mpp = new
                HashMap<>();
mpp.put (xr, 1);
 int cnt = 0;

for( int i = 0; i < n; i++){
   xr = xr ^ a[i];
   int x = xr ^ k;

if (mpp. containKey (x)){
    cnt += mpp.key (x);
 }

if (mpp. Contain key(xr)){
mpp.put (xr, mpp.get(xr)+1);
} else {
   mpp.put (xr, 1);
}
  return cnt;
}.
```

⑦ Merge Overlapping Sub Interval:

[[1,3],[2,6],[8,10],[15,18]]

Output [[1,6],[8,10],[15,18]]

① Current interval [2,6].
Compare ans [1,3]
Overlap check 2<=3? Yes.
Merge [1,3] to [1,6] using Math.max
ans [1,6]. (3,6).

```
for (int i=0; i<n; i++){

if (ans.isEmpty() || arr[i][0]>
  ans.get(ans.size()-1).get(1)){

ans.add(Arrays.asList(arr[i][0],
        arr[i][1]);

else{
ans.get(ans.size()-1).set(1,
Math.max(ans.get(ans.size()-1).get(1),
        arr[i][1]);
}
return ans;
```

⑧ merge two sorted array without Extra Space.
Gap method.

(9) Find the repeating and missing numbers.

$SN = (n*(n+1))/2$

$S^2N = \frac{(n*(n+1)*(2*n+1))}{6}$

Calculate S and S2.

```
long S=0, S2=0.

for(int i=0; i<n; i++){
    S += a[i];
    S2 += (long) a[i]* a[i];
}

long val1 = S-SN;

long val2 = S^2 - S^2N;

val2 = val2 / val1;

long x = (val1 + val2)/2;
long y = x - val1;

int [] ans = {(int) x, (int) y};
return ans;
```

(10). Count Inversion:

$i < j$, $A[j] < A[i]$

N = 5, array[] = {5, 4, 3, 2, 1}

: 10

array[] = {1, 2, 3, 4, 5}

: 0

Merge Sort.

(11). Count Reverse Pair.

N = 5, arr[] = {1,3,2,3,1}

arr[i] > 2*arr[j]

(12) Maximum Product Subarray.

N = [1,2,3,4,5,0]

= 120.

```
int n = arr.length;

int pre=1, suff=1;
int ans = Integer.MIN_Value;

for(int i=0; i<n; i++){
if(pre==0) pre=1;
if(suff==0) suff=1;

pre *= arr[i];
suff *= arr[n-i-1];

ans = Math.max(ans,
        Math.max(pre,suff));
}
return ans;
```