DOMS | Page No.
Date / /

① Largest element in the array

```
int largestElement ( vector<int> & arr, n ) {

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

② Second largest an smallest elements

```
vector<int> getSecondOrderElements (int n, vector<int> a){

    if ( n==0 || n==1 ){
        cout<<-1<<" "<<-1<<endl;
    }
    a.sort ();
    int ssmall = a[1];
    int slargest = a[n-2];
    cout<<slargest<<" "<<small<<endl;
}
```

③ Check if array is sorted and rotated

```
bool check ( vector<int>& nums , n ){
        for ( int i=0; i<n; i++){
                if ( nums[i] < nums[(i-1)] {
                        return false;
                }
        }
        return true;
}
```

④ Remove duplicates from sorted array

```
int removeDuplicates ( vector<int>& arr , n ){
        int i=0;
        for ( int j=1; j<n; j++){
                if ( arr[i] != arr[j]){
                        arr[i+1] = arr[j];
                        i++;
                }
        }
        return i+1;
}
```

⑤ Left rotate an array by one

```
vector<int> rotatearray ( vector<int>& arr, n ){
        int temp = arr[0];
        for (int i=1; i<n; i++){
                arr[i-1] = arr[i];
        }
        arr[n-1] = temp;
```

```
    return arr;
}
```

⑥ Move zero's to end

```
vector<int> movezeros(n, vector<int> a){

    int j=-1;
    for(int i=0; i<n; i++){
        if(a[i]==0){
            j=i;
            break;
        }
    }
    if(j==-1){
        return a;
    }

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

DOMS | Page No.
Date / /

(7) Linear Search

```
int searchInSorted (int arr[], int N, int x){

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

(8) Sort an array of 0's 1's and 2's

```
void sortColors ( vector<int>& nums ) {
    int c1 = 0, c2 = 0, c3 = 0;

    int n = nums.size ();
    for (int i = 0; i < n; i++) {
        if ( nums[i] == 0) {
            c1 ++;
        } else if ( nums[i] == 1) {
            c2 ++;
        } else {
            c3 ++;
        }
    }

    for (int i = 0; i < c1; i++) {
        nums[i] = 0;
    }
    for (int i = c1; i < c1+c2; i++) {
        nums[i] = 1;
    }
```

```
for (int i = c1+c2; i < n; i++){
        nums[i] = 2;
    }
}
```

⑨ Majority element

```
int majorityElement(vector<int> v){

    map<int, int> mpp;
    for (int i=0; i< v.size(); i++){
            mpp[v[i]]++;
    }
    for (auto it: mpp){
            if (it.second > (v.size()/2)){
                return it.first;
            }
    }
    return -1;
}
```

⑩ Kadane's algorithm : Maximum Subarray

```
int maxSubarray(vector<int>& nums){
        long long maxi = LONG_MIN;
        long long sum = 0;

        for (int i=0; i< nums.size(); i++){
            sum += nums[i];
            if (sum > maxi){
                maxi = sum;
            }
```

```
            if ( sum < 0 ){
                sum = 0;
            }
        }
        return maxi;
    }
```

(11) Rearrange the array in alternating positive and negative items.

```
vector <int> rearrangeArray ( vector <int> & nums ) {
    int n = nums.size();
    vector <int> ans (n, 0);
    int pos = 0, neg = 1;
    for ( int i = 0; i < n; i++ ){
        if ( nums[i] < 0 ){
            ans[neg] = nums[i];
            neg += 2;
        } else {
            ans[pas] = nums[i];
            pas += 2;
        }
    }
    return ans;
}
```

(12) Leaders in an array

```
vector <int> leaders ( int n, int arr[] ) {
    vector <int> ans;
    for (int i = 0; i < n; i++ ){
    bool leader = true;
```

```
for (int j = i+1; j < n; j++) {
        if (arr[j] > arr[i]) {
            leader = false;
            break;
        }
    }
}
if (leader) {
    ans.push_back(arr[i]);
}
return ans;
}
```

(13) Rotate matrix by 90 degrees

```
void rotate(vector<vector<int>>& matrix) {
    int n = matrix.size();
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            swap(matrix[i][j], matrix[j][i]);
        }
    }
    for (int i = 0; i < n; i++) {
        reverse(matrix[i].begin(), matrix[i].end());
    }
}
```

DOMS | Page No.
Date / /

(14.) Next permutation

```
void nextPermutation (vector<int>& nums) {
    int n = nums.size(), i = n-2;
    while (i>=0 && nums[i] >= nums[i+1]) {
        i--;
    }
    if (i>=0) {
        int j = n-1;
        while (nums[j] <= nums[i]) {
            j--;
        }
        swap(nums[i], nums[j]);
    }
    reverse(nums.begin() + i+1, nums.end());
}
```

(15.) Count Subarrays with given sum.

```
int subarraySum(vector<int>& nums, int k){
    int n = nums.size();
int cnt = 0;
for(int i=0; i<n; i++){
    int sum = 0;
    for(int j=i; j<n; j++){
        sum += nums[j];

        if(sum == k){
            cnt++;
        }
    }
    return cnt;
}
```

(16) Majority element (n/3 times)

```
vector <int> majorityElement (vector <int> v) {
    int n = v.size();
    vector <int> ls;

    for (int i=0; i<n; i++){
        if (ls.size() == 0 || ls[0] != v[i]) {
            int cnt = 0;
            for (int j=0; j<n; j++){
                if (v[j] == v[i]){
                    cnt++;
                }
            }
            if (cnt > (n/3)){
                ls.push_back (v[i]);
            }
            if (ls.size() == 2) {
                break;
            }
    }
    return ls;
}
```

(17.) Merge overlapping Subintervals

```
vector <vector <int>> mergeOver (vector <vector <int>> & arr){

    int n = arr.size();
    Sort (arr.begin(), arr.end());
    vector <vector <int>> ans;
```

```
for (int i = 0; i < n ; i++) {
    int start = arr[i][0];
    int end = arr[i][1];

    if (!ans.empty() && end <= ans.back()[1]) {
        continue;
    }

    for (int j = i+1; j < n; j++){
        if ( arr[j][0] <= end ){
            end = max(end, arr[j][1]);
        } else {
            break;
        }
    }
    ans.push_back({start, end});
}
return ans;
}
```

(18) Finding missing and repeating numbers

```
vector <int> findMissingRepeating (vector <int> a){

int n = a.size();

int repeat = -1, missing = -1;
for (int i = 1; i <= n; i++){
    int c=0;
    for (int j=0; j < n; j++) {
        if (a[j] == i){
            c++;
```

```
        }
    }
    if (c == 2) {
        repeat = i;
    } else if (c == 0) {
        missing = i;
    }
    if (repeat != -1 && missing != -1) {
        break;
    }
}
return { repeat, missing };
}
```

(19) Count inversions

```
int numberOfInversions (vector<int>& a, int n) {

    int count = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            if (a[i] > a[j]) {
                count++;
            }
        }
    }
    return count;
}
```

(20) Reverse pairs

```
int team (vector<int>& skill, int n){
    int count = 0;
    for (int i = 0; i < n; i++){
        for (int j = i+1; j < n; j++){
            if (skill[i] > 2 * skill[j]){
                count++;
            }
        }
    }
    return count;
}
```

(21) Maximum product Subarray

```
int Subarray (vector<int>& arr){
    int result = arr[0];
    for (int i=0; i < arr.size()-1; i++){
        int p = arr[i];
        for (int j = i+1; j < arr.size(); j++){
            result = max(result, p);
            p *= arr[j];
        }
        result = max(result, p);
    }
    return result;
}
```