DAY 1 -> Array 1

Dutch National Flag

```
void sortColors(vector<int>& nums) {
       int low = 0, mid = 0, high = nums.size() - 1;
       while(mid <= high){
           if(nums[mid] == 1){
               mid ++;
           }else if(nums[mid] == 0){
               swap(nums[mid], nums[low]);
               mid ++; low++;
           }else if(nums[mid] == 2){
               swap(nums[mid], nums[high]);
               high--;
           }
       }
   }
```

Max Subarray Sum (**Kadane's Algorithm**)

```
int maxSubArray(vector<int>& nums) {
       int n = nums.size();
       int maxi = INT_MIN, sum = 0;
       for(int i = 0; i < n; i++){
           sum = max(nums[i], sum + nums[i]);
           maxi = max(maxi, sum);
       }
       return maxi;
   }
```

Next Permutation

```
void nextPermutation(vector<int>& nums) {
       int pivot = -1, n = nums.size();
       for(int i = n - 2; i >= 0; i--){
```

```
            if(nums[i] < nums[i + 1]){
                pivot = i;
                break;
            }
        }
        if(pivot == -1){
            reverse(nums.begin(), nums.end());
            return;
        }
        for(int i = nums.size() - 1; i >= pivot; i--){
            if(nums[i] > nums[pivot]){
                swap(nums[i], nums[pivot]);
                break;
            }
        }
        reverse(nums.begin() + pivot + 1, nums.end());
    }
```

Max Buy and sell

```
int maxProfit(vector<int>& prices) {
        int buy = prices[0], profit = 0, sell = INT_MIN;
        for(int i = 0; i < prices.size() - 1; i++){
            if(prices[i] < prices[i + 1]){
                buy = min(buy, prices[i]);
                sell = max(prices[i + 1] - buy, sell);
            }
        }
        return sell != INT_MIN ? sell : 0;
    }
```

Pascal Triangle

```
int mod = 1e9 + 7;
    vector<vector<int>> generate(int n) {
        vector<vector<int>> answer(n);
        for(int i = 0; i < n; i++){
            answer[i].resize(i + 1);
```

```
            answer[i][0] = answer[i][i] = 1;
            for(int j = 1; j < i; j++){
                answer[i][j] = (answer[i - 1][j - 1] + answer[i - 1][j]) % mod;
            }
        }
        return answer;
    }
```