# ARRAYS

## 1. INSERT

```
#include <iostream>
#include <cmath>
using namespace std;

int insert(int arr[], int n, int x, int cap, int pos)
{
        if(n == cap)
                return n;

        int idx = pos - 1;

        for(int i = n - 1; i >= idx; i--)
        {
                arr[i + 1] = arr[i];
        }

        arr[idx] = x;

        return n + 1;
}



int main() {

      int arr[5], cap = 5, n = 3;

      arr[0] = 5; arr[1] = 10; arr[2] = 20;
```

```
    cout<<"Before Insertion"<<endl;

    for(int i=0; i < n; i++)
    {
      cout<<arr[i]<<" ";
    }

    cout<<endl;

    int x = 7, pos = 2;

    n = insert(arr, n, x, cap, pos);

    cout<<"After Insertion"<<endl;

    for(int i=0; i < n; i++)
    {
            cout<<arr[i]<<" ";
    }

}
```

## 2. SEARCH

```
#include <iostream>
#include <cmath>
using namespace std;

int search(int arr[], int n, int x)
{
    for(int i = 0; i < n; i++)
    {
        if(arr[i] == x)
            return i;
```

```
        }

        return -1;
}



int main() {

        int arr[] = {20, 5, 7, 25}, x = 5;

    cout<<search(arr, 4, x);

}
```

### 3. DELETION

```
#include <iostream>
#include <cmath>
using namespace std;

int deleteEle(int arr[], int n, int x)
{
        int i = 0;

        for(i = 0; i < n; i++)
        {
                if(arr[i] == x)
                        break;
        }

        if(i == n)
                return n;
```

```
        for(int j = i; j < n - 1; j++)
        {
                arr[j] = arr[j + 1];
        }

        return n-1;
}


int main() {

        int arr[] = {3, 8, 12, 5, 6}, x = 12, n = 5;

        cout<<"Before Deletion"<<endl;

        for(int i=0; i < n; i++)
        {
          cout<<arr[i]<<" ";
        }

        cout<<endl;


        n = deleteEle(arr, n, x);

        cout<<"After Deletion"<<endl;

        for(int i=0; i < n; i++)
        {
                cout<<arr[i]<<" ";
        }

}
```

## 4. SECOND LARGEST ELEMENT

```
#include <stdio.h>
int secondlargest(int a[],int n)
{
    int largest=0;
    int res=-1;
    for(int i=1;i<n;i++)
    {
        if(a[i]>a[largest])
        {
            res=largest;
            largest=i;
        }
        else if(a[i]!=a[largest])
        {
            if(res==-1||a[i]>a[res])
            res=i;
        }
    }
    return res;
}
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    int pos=secondlargest(a,n);
    printf("%d ",a[pos]);
return 0;
}
```

## 5. CHECK IF ARRAY IS SORTED

```
#include <iostream>
#include <cmath>
using namespace std;

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

        return true;
}


int main() {

     int arr[] = {5, 12, 30, 2, 35}, n = 5;

     printf("%s", isSorted(arr, n)? "true": "false");

}
```

## 6. REVERSE AN ARRAY

```
#include <iostream>
#include <cmath>
using namespace std;

 void reverse(int arr[], int n)
{
```

```
        int low = 0, high = n - 1;

        while(low < high)
        {
                int temp = arr[low];

                arr[low] = arr[high];

                arr[high] = temp;

                low++;
                high--;
        }
}


int main() {

      int arr[] = {10, 5, 7, 30}, n = 4;

      cout<<"Before Reverse"<<endl;

       for(int i = 0; i < n; i++)
       {
                cout<<arr[i]<<" ";
       }

       cout<<endl;

       reverse(arr, n);

       cout<<"After Reverse"<<endl;

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

```
            cout<<arr[i]<<" ";
        }

    }
```

## 7. REMOVE DUPLICATES FROM SORTED ARRAY

```
#include <iostream>
#include <cmath>
using namespace std;

 int remDups(int arr[], int n)
{
        int res = 1;

        for(int i = 1; i < n; i++)
        {
                if(arr[res - 1] != arr[i])
                {
                        arr[res] = arr[i];
                        res++;
                }
        }

        return res;
}

int main() {

      int arr[] = {10, 20, 20, 30, 30, 30}, n = 6;

      cout<<"Before Removal"<<endl;

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

```
	{
		cout<<arr[i]<<" ";
	}

	cout<<endl;

	n = remDups(arr, n);

	cout<<"After Removal"<<endl;

	for(int i = 0; i < n; i++)
	{
		cout<<arr[i]<<" ";
	}

}
```

## 8. LEFT ROTATE AN ARRAY BY ONE

```
#include <iostream>
#include <cmath>
using namespace std;

 void lRotateOne(int arr[], int n)
{
	int temp = arr[0];

	for(int i = 1; i < n; i++)
	{
		arr[i - 1] = arr[i];
	}

	arr[n - 1] = temp;
}
```

```
int main() {

	int arr[] = {1, 2, 3, 4, 5}, n = 5;

	cout<<"Before Rotation"<<endl;

	for(int i = 0; i < n; i++)
	{
		cout<<arr[i]<<" ";
	}

	cout<<endl;

	lRotateOne(arr, n);

	cout<<"After Rotation"<<endl;

	for(int i = 0; i < n; i++)
	{
		cout<<arr[i]<<" ";
	}

}
```

## 9. LEFT ROTATE AN ARRAY BY D PLACES

```
#include <iostream>
#include <cmath>
using namespace std;


void reverse(int arr[], int low, int high)
{
```

```
        while(low < high)
        {
                swap(arr[high], arr[low]);

                low++;
                high--;
        }
}

void leftRotate(int arr[], int d, int n)
{
        reverse(arr, 0, d - 1);

        reverse(arr, d, n - 1);

        reverse(arr, 0, n - 1);
}



int main() {

      int arr[] = {1, 2, 3, 4, 5}, n = 5, d = 2;

      cout<<"Before Rotation"<<endl;

       for(int i = 0; i < n; i++)
       {
                cout<<arr[i]<<" ";
       }

       cout<<endl;

       leftRotate(arr, d, n);
```

```
        cout<<"After Rotation"<<endl;

        for(int i = 0; i < n; i++)
        {
                cout<<arr[i]<<" ";
        }

}
```

## 10. LEADERS IN AN ARRAY

```
#include <iostream>
#include <cmath>
using namespace std;


void leaders(int arr[], int n)
{
        int curr_ldr = arr[n - 1];

        cout<<curr_ldr<<" ";

        for(int i = n - 2; i >= 0; i--)
        {
                if(curr_ldr < arr[i])
                {
                        curr_ldr = arr[i];

                        cout<<curr_ldr<<" ";
                }
        }
}
```

```
int main() {

	int arr[] = {7, 10, 4, 10, 6, 5, 2}, n = 7;

	leaders(arr, n);

}
```

## 11. MAXIMUM DIFFERENCE PROBLEM WITH ORDER

```
#include <iostream>
#include <cmath>
using namespace std;


int maxDiff(int arr[], int n)
{
	int res = arr[1] - arr[0], minVal = arr[0];

	for(int i = 1; i < n; i++)
	{

		res = max(res, arr[i] - minVal);

		minVal = min(minVal, arr[i]);
	}

	return res;
}



int main() {
```

```
    int arr[] = {2, 3, 10, 6, 4, 8, 1}, n = 7;

    cout<<maxDiff(arr, n);

}
```

## 12. FREQUENCIES IN A SORTED ARRAY

```
#include <iostream>
#include <cmath>
using namespace std;


void printFreq(int arr[], int n)
{
        int freq = 1, i = 1;

        while(i < n)
        {
                while(i < n && arr[i] == arr[i - 1])
                {
                        freq++;
                        i++;
                }

                cout<<arr[i - 1] << " " << freq << endl;

                i++;
                freq = 1;
        }
}
```

```
int main() {

    int arr[] = {10, 10, 20, 30, 30, 30}, n = 6;

    printFreq(arr, n);

}
```

## 13. STOCK BUY AND SELL

```
#include <iostream>
#include <cmath>
using namespace std;


int maxProfit(int price[], int n)
{
        int profit = 0;

        for(int i = 1; i < n; i++)
        {
                if(price[i] > price[i - 1])
                        profit += price[i] - price[i -1];
        }

        return profit;

}


int main() {

    int arr[] = {1, 5, 3, 8, 12}, n = 5;
```

```
    cout<<maxProfit(arr, n);

}
```

**14.**

**15. TRAPPING RAIN WATER**

```
#include <iostream>
#include <cmath>
using namespace std;


int getWater(int arr[], int n)
{
        int res = 0;

        for(int i = 1; i < n - 1; i++)
        {
                int res = 0;

        int lMax[n];
        int rMax[n];

        lMax[0] = arr[0];
        for(int i = 1; i < n; i++)
                lMax[i] = max(arr[i], lMax[i - 1]);


        rMax[n - 1] = arr[n - 1];
        for(int i = n - 2; i >= 0; i--)
                rMax[i] = max(arr[i], rMax[i + 1]);

        for(int i = 1; i < n - 1; i++)
                res = res + (min(lMax[i], rMax[i]) - arr[i]);
```

```
        return res;
        }

        return res;
}


int main() {

    int arr[] = {5, 0, 6, 2, 3}, n = 5;

    cout<<getWater(arr, n);

}
```

## 16. MAXIMUM CONSECUTIVE ONES

```
#include <iostream>
#include <cmath>
using namespace std;


int maxConsecutiveOnes(int arr[], int n)
{
        int res = 0, curr = 0;

        for(int i = 0; i < n; i++)
        {
                if(arr[i] == 0)
                        curr = 0;
                else
                {
                        curr++;
```

```
                res = max(res, curr);
            }
        }

        return res;
}



int main() {

    int arr[] = {0, 1, 1, 0, 1, 1, 1}, n = 7;

    cout<<maxConsecutiveOnes(arr, n);

}
```

## 17. MAXIMUM SUBARRAY SUM

```
#include <iostream>
#include <cmath>
using namespace std;


int maxSum(int arr[], int n)
{
        int res = arr[0];

        int maxEnding = arr[0];

        for(int i = 1; i < n; i++)
        {
                maxEnding = max(maxEnding + arr[i], arr[i]);
```

```
            res = max(maxEnding, res);
        }

        return res;
}



int main() {

    int arr[] = {1, -2, 3, -1, 2}, n = 5;

    cout<<maxSum(arr, n);

}
```

## 18. LONGEST EVEN ODD SUBARRAY

```
#include <iostream>
#include <cmath>
using namespace std;


int maxEvenOdd(int arr[], int n)
{
        int res = 1;
        int curr = 1;
        for(int i = 1; i < n; i++)
        {
                if((arr[i] % 2 == 0 && arr[i - 1] % 2 != 0)
                  ||(arr[i] % 2 != 0 && arr[i - 1] % 2 == 0))
                        {
                                curr++;
```

```
                        res = max(res, curr);
                    }
                    else
                        curr = 1;
        }
        return res;
}
int main() {

    int arr[] = {5, 10, 20, 6, 3, 8}, n = 6;

    cout<<maxEvenOdd(arr, n);

}
```

## 19. MAXIMUM CIRCULAR SUM SUBARRAY

```
#include <iostream>
#include <cmath>
using namespace std;


int normalMaxSum(int arr[], int n)
{
        int res = arr[0];

        int maxEnding = arr[0];

        for(int i = 1; i < n; i++)
        {
                maxEnding = max(maxEnding + arr[i], arr[i]);

                res = max(maxEnding, res);
        }
```

```
        return res;
}

int overallMaxSum(int arr[], int n)
{
        int max_normal = normalMaxSum(arr, n);

        if(max_normal < 0)
                return max_normal;

        int arr_sum = 0;

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

                arr[i] = -arr[i];
        }

        int max_circular = arr_sum + normalMaxSum(arr, n);

        return max(max_circular, max_normal);
}



int main() {

    int arr[] = {8, -4, 3, -5, 4}, n = 5;

    cout<<overallMaxSum(arr, n);

}
```

## 20. MAJORITY ELEMENT

```
#include <iostream>
#include <cmath>
using namespace std;


int findMajority(int arr[], int n)
{
	int res = 0, count = 1;

	for(int i = 1; i < n; i++)
	{
		if(arr[res] == arr[i])
			count++;
		else
			count --;

		if(count == 0)
		{
			res = i; count = 1;
		}
	}

	count = 0;

	for(int i = 0; i < n; i++)
		if(arr[res] == arr[i])
			count++;

	if(count <= n /2)
		res = -1;

	return res;
```

```
}


int main() {

    int arr[] = {8, 8, 6, 6, 6, 4, 6}, n = 7;

    cout<<findMajority(arr, n);

}
```

## 21. MINIMUM CONSECUTIVE FLIPS

```
#include <iostream>
#include <cmath>
using namespace std;


void printGroups(int arr[], int n)
{
        for(int i = 1; i < n; i++)
        {
                if(arr[i] != arr[i - 1])
                {
                        if(arr[i] != arr[0])
              cout << "From " << i << " to ";
                        else
              cout << i - 1 << endl;
                }
        }

        if(arr[n - 1] != arr[0])
       cout << n - 1 << endl;
```

```
}


int main() {
    int arr[] = {0, 0, 1, 1, 0, 0, 1, 1, 0}, n = 9;
    printGroups(arr, n);
}
```

## 22. Maximum Sum of K Consecutive elements

```
#include <iostream>
#include <cmath>
#include <climits>
using namespace std;


int maxSum(int arr[], int n, int k)
{
        int curr_sum = 0;

        for(int i = 0; i < k; i++)
                curr_sum += arr[i];

        int max_sum = curr_sum;

        for(int i = k; i < n; i++)
        {
                curr_sum += (arr[i] - arr[i - k]);

                max_sum = max(max_sum, curr_sum);
```

```
        }

        return max_sum;
}


int main() {

    int arr[] = {1, 8, 30, -5, 20, 7}, n = 6, k = 3;

    cout<<maxSum(arr, n, k);

}
```

## 23. Find subarray with given sum

```
/* An efficient program to print subarray with sum as given sum */
#include<stdio.h>

/* Returns true if the there is a subarray of arr[] with a sum equal to 'sum'
otherwise returns false. Also, prints the result */
int subArraySum(int arr[], int n, int sum)
{
        /* Initialize curr_sum as value of first element
        and starting point as 0 */
        int curr_sum = arr[0], start = 0, i;

        /* Add elements one by one to curr_sum and if the curr_sum exceeds the
        sum, then remove starting element */
        for (i = 1; i <= n; i++)
        {
                // If curr_sum exceeds the sum, then remove the starting elements
                while (curr_sum > sum && start < i-1)
                {
```

```
                curr_sum = curr_sum - arr[start];
                start++;
            }

            // If curr_sum becomes equal to sum, then return true
            if (curr_sum == sum)
            {
                printf ("Sum found between indexes %d and %d", start, i-1);
                return 1;
            }

            // Add this element to curr_sum
            if (i < n)
            curr_sum = curr_sum + arr[i];
        }

        // If we reach here, then no subarray
        printf("No subarray found");
        return 0;
}

// Driver program to test above function
int main()
{
        int arr[] = {15, 2, 4, 8, 9, 5, 10, 23};
        int n = sizeof(arr)/sizeof(arr[0]);
        int sum = 23;
        subArraySum(arr, n, sum);
        return 0;
}
```

## 24. N-bonacci numbers

```
// CPP program print first M terms of
```

```
// N-bonacci series.
#include <bits/stdc++.h>
using namespace std;

// Function to print bonacci series
void bonacciseries(long n, int m)
{

	// Assuming m > n.
	int a[m] = { 0 };
	a[n - 1] = 1;
	a[n] = 1;

	// Uses sliding window
	for (int i = n + 1; i < m; i++)
		a[i] = 2 * a[i - 1] - a[i - n - 1];

	// Printing result
	for (int i = 0; i < m; i++)
		cout << a[i] << " ";
}

// Driver's Code
int main()
{
	int N = 5, M = 15;
	bonacciseries(N, M);
	return 0;
}
```

## 25. PREFIX SUM

```
#include <iostream>
#include <cmath>
#include <climits>
```

```
using namespace std;


int prefix_sum[10000];

void preSum(int arr[], int n)
{


	prefix_sum[0] = arr[0];

	for(int i = 1; i < n; i++)
	{
		prefix_sum[i] = prefix_sum[i - 1] + arr[i];
	}


}

int getSum(int prefix_sum[], int l, int r)
{
	if(l != 0)
		return prefix_sum[r] - prefix_sum[l - 1];
	else
		return prefix_sum[r];
}



int main() {

    int arr[] = {2, 8, 3, 9, 6, 5, 4}, n = 7;

    preSum(arr, n);
```

```
    cout<<getSum(prefix_sum, 1, 3)<<endl;

    cout<<getSum(prefix_sum, 0, 2)<<endl;

}
```

## 26. EQUILIBRIUM POINT

```
#include <iostream>
#include <cmath>
#include <climits>
using namespace std;


bool checkEquilibrium(int arr[], int n)
{
        int sum = 0;

        for(int i = 0; i < n; i++)
        {
                sum += arr[i];
        }

        int l_sum = 0;

        for(int i = 0; i < n; i++)
        {
                if(l_sum == sum - arr[i])
                        return true;

                l_sum += arr[i];

                sum -= arr[i];
        }
```

```
        return false;
}



int main() {

    int arr[] = {3, 4, 8, -9, 20, 6}, n = 6;

    printf("%s",checkEquilibrium(arr, n)? "true" : "false");


}
```

## 27. MAXIMUM OCCURRING ELEMENT

```
#include <iostream>
#include <cmath>
#include <bits/stdc++.h>
#include <climits>
using namespace std;


int maxOcc(int L[], int R[], int n)
{
        int arr[1000];

        memset(arr, 0, sizeof(arr));

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

```
            arr[R[i] + 1]--;
        }

        int maxm = arr[0], res = 0;

        for(int i = 1; i < 1000; i++)
        {
            arr[i] += arr[i - 1];

            if(maxm < arr[i])
            {
                maxm = arr[i];

                res = i;
            }
        }

        return res;
}



int main() {

    int L[] = {1, 2, 3}, R[] = {3, 5, 7}, n = 3;

    cout<<maxOcc(L, R, n);


}
```