DATA STRUCTURES & ALGORITHMS

PRACTICE
Week 1 - Search

Lecture: MCS. Mai Thai Quoc
Email: maithaiquoc@gmail.com

POPULARIS
GIA DINH
UNIVERSITY

PRACTICE 1

Binary search (no recursion)

```
#include <iostream>
using namespace std;
int a[] = {2, 5, 7, 9, 4, 8, 20};
int n = sizeof(a)/sizeof(a[0]);
int binarySearch(int el, int left, int right){
        int mid= left;
        while (right >=left) {
                mid = (left+right)/2;
                if(el==a[mid]) return mid;
                else if(a[mid]<el) left = mid + 1;
                else right = mid-1;
        }
        return -1;
}
```

```
void main(){
cout<<binarySearch(15,0,n)<<endl;
cout<<binarySearch(20,0,n)<<endl;
}
```

PRACTICE 2

Convert n from decimal to binary, print the result.

```
#include <iostream>
using namespace std;
void decToBinary(int n)
{
    int binaryNum[32];
    int i = 0;
    while (n > 0) {
        binaryNum[i] = n % 2;
        n = n / 2;
        i++;
    }
    for (int j = i - 1; j >= 0; j--)
        cout << binaryNum[j];
}
```

```
int main(){
    int n = 20;
    decToBinary(n);
    return 0;
}
```

PRACTICE 3

Given an array of integers, find sum of its elements.

Examples :

Input : arr[] = {1, 2, 3}

Output : 6

1 + 2 + 3 = 6

Input : arr[] = {15, 12, 13, 10}

Output : 50

PRACTICE 3

```
using namespace std;
#include<iostream>
int sum(int arr[], int n)  {
    int sum = 0;
    for (int i = 0; i < n; i++)
       sum += arr[i];
    return sum;
}
```

```
int main(){
int arr[] = {12, 3, 4, 15};
int n = sizeof(arr) / sizeof(arr[0]);
cout << "Sum of given array is " <<
sum(arr, n);
return 0;
}
```

PRACTICE 4

Given an array of n elements, the task is to find the elements that are greater than half of elements in an array. In case of odd elements, we need to print elements larger than floor(n/2) elements where n is total number of elements in array.

Examples :

Input : arr[] = {1, 6, 3, 4}

Output : 4 6

Input : arr[] = {10, 4, 2, 8, 9}

Output : 10 9 8

PRACTICE 4

```
#include <bits/stdc++.h>
using namespace std;
#include<iostream>
using namespace std;
void findLarger(int arr[], int n)
{
    sort(arr, arr + n);
    for (int i = n-1; i >= n/2; i--)
        cout << arr[i] << " ";
}
```

```
int main(){
        int arr[] = {1, 3, 6, 1, 0, 9};
        int n = sizeof(arr)/sizeof(arr[0]);
        findLarger(arr, n);
        return 0;
}
```

PRACTICE 5

Given an array of n distinct elements, the task is to find all elements in array which have at-least two greater elements than themselves.

Examples :

Input : arr[] = {2, 8, 7, 1, 5};

Output : 2 1 5

The output three elements have two or
more greater elements

Input : arr[] = {7, -2, 3, 4, 9, -1};

Output : -2 3 4 -1

PRACTICE 5

```
using namespace std;
#include<iostream>
void findElements(int arr[], int n) {
    for (int i = 0; i < n; i++)     {
        int count = 0;
        for (int j = 0; j < n; j++)
            if (arr[j] > arr[i])
                count++;
        if (count >= 2)
            cout << arr[i] << " ";
    }
}
```

```
int main(){
    int arr[] = { 2, -6 ,3 , 5, 1};
    int n = sizeof(arr) / sizeof(arr[0]);
    findElements(arr, n);
    return 0;
}
```

PRACTICE 5

```
#include <bits/stdc++.h>
using namespace std;
#include<iostream>
void findElements(int arr[], int n)
{
    sort(arr, arr + n);

    for (int i = 0; i < n - 2; i++)
    cout << arr[i] << " ";
}
```

```
int main(){
    int arr[] = { 2, -6 ,3 , 5, 1};
    int n = sizeof(arr) / sizeof(arr[0]);
    findElements(arr, n);
    return 0;
}
```

PRACTICE 5

```
using namespace std;
#include<iostream>
void findElements(int arr[], int n) {
    int first = INT_MIN, second = INT_MIN;
    for (int i = 0; i < n; i++) {
        if (arr[i] > first) {
            second = first;
            first = arr[i];
        }
       else if (arr[i] > second)
              second = arr[i];
    }
    for (int i = 0; i < n; i++)
        if (arr[i] < second)
            cout << arr[i] << " ";
}
```

```
int main(){
    int arr[] = { 2, -6, 3, 5, 1};
    int n = sizeof(arr) / sizeof(arr[0]);
    findElements(arr, n);
    return 0;
}
```

PRACTICE 6

Given the length of an array of integers **N** and an integer **K**. The task is to modify the array in such a way that the array contains first all odd integers from 1 to **N** in ascending order, then all even integers from 1 to **N** in ascending order and then print the **K**th element in the modified array.

Examples:

Input: N = 8, K = 5

Output: 2

The array will be {1, 3, 5, 7, 2, 4, 6, 8}

and the fifth element is 2.

PRACTICE 6

```
#include <bits/stdc++.h>
using namespace std;
#include<iostream>
int getNumber(int n, int k){
    int arr[n];
    int i = 0;
    int odd = 1;
    while (odd <= n) {
        arr[i++] = odd;
        odd += 2;
    }
    int even = 2;
    while (even <= n) {
            arr[i++] = even;
            even += 2;
    }
    return arr[k - 1];
}

int main(){
      int n = 8, k = 5;
      cout << getNumber(n, k);
      return 0;
}
```

PRACTICE 6

```
#include <bits/stdc++.h>
using namespace std;
#include<iostream>
int getNumber(int n, int k) {
    int pos;
    if (n % 2 == 0) {
        pos = n / 2;
    }
    else {
        pos = (n / 2) + 1;
    }
    if (k <= pos)
        return (k * 2 - 1);
    else
        return ((k - pos) * 2);
}
```

```
int main(){
        int n = 8, k = 5;
        cout << getNumber(n, k);
        return 0;
}
```

PRACTICE 7

Find a pair with maximum product in array of Integers

Given an array with both +ive and -ive integers, return a pair with highest product.

Examples :

Input: arr[] = {1, 4, 3, 6, 7, 0}

Output: {6,7}

Input: arr[] = {-1, -3, -4, 2, 0, -5}

Output: {-4,-5}

PRACTICE 7

```
#include <iostream>
using namespace std;
void maxProduct(int arr[], int n){
    if (n < 2)
    {
        cout << "No pairs exists\n";
        return;
    }
   int a = arr[0], b = arr[1];
    for (int i=0; i<n; i++)
      for (int j=i+1; j<n; j++)
         if (arr[i]*arr[j] > a*b)
            a = arr[i], b = arr[j];
    cout << "Max product pair is {" << a <<
", "<< b << "}";
}
```

```
int main(){
    int arr[] = {1, 4, 3, 6, 7, 0};
    int n = sizeof(arr)/sizeof(arr[0]);
    maxProduct(arr, n);
    return 0;
}
```