#include <bits/stdc++.h>

using namespace std;

const int N = 10;

int a[N];

void in(int a[])

{

    for (int i = 0; i < N; i++)

        a[i] = rand();

}

void userInput(int a[], int n)

{

    for (int i = 0; i < n; i++)

    {

        cin >> a[i];

    }

}

void print(int a[], int n)

{

    for (int i = 0; i < n; i++)

        cout << a[i] << " ";

    cout << endl;

}

int linear(int a[],int n,int ind,int k)

{

    if (ind>=n){

        return -1;

    }

    if(a[ind]!=k)linear(a,n,ind+1,k);

    else return ind;

}

int main()

{

    //    in(a);

    int n, k;

    cout << "enter n:";

    cin >> n;

    userInput(a, n);

    print(a, n);

    cout << "enter num to search:";

    cin >> k;

    // iterative linear search

    // tc:o(n)

    // sc: o(1) no extra space req

    int flag = -1, ind = -1;

    for (int i = 0; i < n; i++)

    {

        if (a[i] == k)

        {

            flag = 1;

            ind = i;

            break;

        }

    }

    if (flag == 1)

        cout << k << " found at index : " << ind << endl;

    else

        cout << "not found";

    // recursive

    // sc:o(n) worst and avg case

    // tc:o(n) avg case

    int ans=linear(a,n,0,k);

    if(ans==-1)cout<<"not found\n";

    else cout<<"found at "<<ans<<endl;

}

#include <bits/stdc++.h>

using namespace std;

const int N = 10;

int a[N];

void in(int a[])

{

    for (int i = 0; i < N; i++)

        a[i] = rand();

}

void userInput(int a[], int n)

{

    for (int i = 0; i < n; i++)

    {

        cin >> a[i];

    }

}

void print(int a[], int n)

{

    for (int i = 0; i < n; i++)

        cout << a[i] << " ";

    cout << endl;

}

int fib(int n)

{

    if (n == 0 || n == 1)

        return n;

    return fib(n - 1) + fib(n - 2);

}

int main()

{

    //    in(a);

    int n, k;

    cout << "enter n:";

    cin >> n;

    cout << "nth element is :" << fib(n - 1) << endl;

    // iterative fib

    // sc:o(n) if storing in array

    // tc:o(n);

    int fibo[n];

    for (int i = 0; i < n; i++)

    {

        if (i == 0 || i == 1)

            fibo[i] = i;

        else

            fibo[i] = fibo[i - 1] + fibo[i - 2];

    }

    //    print(fibo,n);

    // recursive

    // tc:o(2^n), sc:o(n)

    cout << "nth element is :" << fibo[n - 1] << endl;

}

#include<bits/stdc++.h>

using namespace std;

void merge(int arr[], int left, int mid, int right) {  // tc: O(n)

// sc: O(N)

    int n1 = mid - left + 1;

    int n2 = right - mid;

    int L[n1+1];

    int R[n2+1];

    L[n1]=1e9;

    R[n2]=1e9;

    for (int i = 0; i < n1; i++)

        L[i] = arr[left + i];

    for (int j = 0; j < n2; j++)

        R[j] = arr[mid + 1 + j];

    int i = 0, j = 0, k = left;

    while (i < n1 && j < n2) {

        if (L[i] <= R[j]) {

            arr[k] = L[i];

            i++;

        } else {

            arr[k] = R[j];

            j++;

        }

        k++;

    }

}

void mergeRec(int arr[], int left, int right) {

    // sc: O(n+log(n))

    // tc: O(nlogn)

    if (left < right) {

        int mid = left + (right - left) / 2;

        mergeRec(arr, left, mid);

        mergeRec(arr, mid + 1, right);

        merge(arr, left, mid, right);

    }

}

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

    // sc:o(n)

    // tc:o(nlog(n))

    for (int i = 1; i <= n - 1; i = 2 \*i) {

        for (int left = 0; left < n - 1; left += 2 \* i) {

            int mid = min(left + i - 1, n - 1);

            int right = min(left + 2 \* i - 1, n - 1);

            merge(arr, left, mid, right);

        }

    }

}

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

    // tc: o(n)

    for(int i=0;i<n;i++){

        cout<<arr[i]<<" ";

    }

    cout<<endl;

}

int main(){

    int n;

    cout<<"size:";

    cin>>n;

    int arr[n];

    int x;

    for(int i=0;i<n;i++){

        cin>>arr[i];

    }

    print(arr,n);

    cout<<"\n";

    //recursive

    mergeRec(arr,0,n-1);   //sc: o(n+log(n)), tc:o(n\*logn)

    print(arr,n);

    cout<<"\n";

    // iterative

    mergeSort(arr,n);  // sc:o(n) tc: o(nlogn)

    print(arr,n);

    // mergeSortRecursive(arr,0,n);

}