1. **Given an array arr[] of integers. Find a peak element i.e. an element that is not smaller than its neighbors.**

Sample Input: arr[] = { 5, 10,20,15}

Sample Output: 20

Explanation: The element 20 has neighbours 10 and 15, both of them are less than 20.

Note: For corner elements, we need to consider only one neighbour. If all elements of the input array are the same, every element is a peak element.

**Code:**

//21161 Shaik Nazeer CSE-B

#include<bits/stdc++.h>

#define ll long long

#define loop(i,n) for(int i = 0; i < n; i++)

#define loop1(i,n) for(int i = 1; i <= n; i++)

using namespace std;

int main()

{

    int n,i=0;

    cin>>n;

    int a[n];

    loop(i,n){

        cin>>a[i];

    }

    int ans=a[0];

    if(n==1) {

        cout<<a[0]<<endl;

    }

    else if(a[0]>a[1]){

        cout<<a[1]<<endl;

    }

    else if(a[n-1]>a[n-2]){

        cout<<a[n-1]<<endl;

    }

    else{

        for(i = 1; i < n-1; i++){

            if(a[i]>a[i-1] && a[i]>a[i+1]){

                cout<<a[i]<<endl;

                return 0;

            }

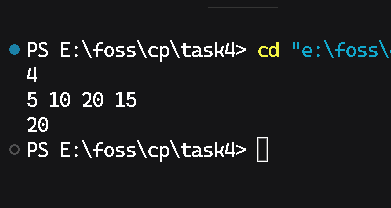
        }

    }

    return 0;

}

**Screenshot:**



1. **Given an array A[] consisting of only 0s, 1s, and 2s. The task is to write a function that sorts the given array. The functions should put all 0s first, then all 1s and all 2s in last.**

Sample Input: A[] = {0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1}.

Sample Output: {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2}.

Constraints: The expected time complexity is O(N). You can’t use any inbuilt sorting functions.

**Code:**

*//21161 Shaik Nazeer CSE-B*

#include<bits/stdc++.h>

#define ll long long

#define loop(*i*,*n*) for(int i = 0; i < n; i++)

#define loop1(*i*,*n*) for(int i = 1; i <= n; i++)

using namespace std;

int main()

{

    int n;

    cin>>n;

    int a[n];

    loop(i,n) cin>>a[i];

    int i = 0, j = n-1,k = 0;

    while(j>0 && a[j]==2) j--;

    while(i<n && a[i]==0) i++;

    while(i<j && k< n) {

        if(a[k]==0 && i<k){

            swap(a[i],a[k]);

            i++;

        }else if(a[k]==2 && k<j){

            swap(a[j],a[k]);

            j--;

        }else k++;

    }

    loop(i,n) cout<<a[i]<<" ";

    return 0;

}

**Screenshot:**

**A screen shot of a computer

Description automatically generated with medium confidence**

1. **Find the majority element in the array. A majority element in an array A[] of size n is an element that appears more than n/2 times.**

Input : A[]={3, 3, 4, 2, 4, 4, 2, 4,4}

Output : 4

Explanation: The frequency of 4 is 5 which is greater than the half of the size of the array.

Constraints: The expected time complexity is O(N).

**Code:**

//21161 Shaik Nazeer CSE-B

#include<bits/stdc++.h>

#define ll long long

#define loop(i,n) for(int i = 0; i < n; i++)

#define loop1(i,n) for(int i = 1; i <= n; i++)

using namespace std;

int main()

{

    int n,cnt=1;

    cin>>n;

    int a[n];

    loop(i,n) cin>>a[i];

    sort(a,a+n);

    loop(i,n-1) {

        if(a[i]==a[i+1]){

            cnt++;

        }else {

            if(cnt>n/2){

                cout<<a[i-1]<<endl;

                return 0;

            }

            cnt=1;

        }

    }

    cout<<a[n-1]<<endl;

    return 0;

}

**Screenshot:**

**A picture containing text, font, screenshot

Description automatically generated**

1. **Given an array arr[] of size N, the task is to sort this array in descending order.**

Input: arr[] = {0, 23, 14, 12, 9}

Output: {23, 14, 12, 9, 0}

Constraints: Don’t use any inbuilt functions.

**Code:**

*//21161 Shaik Nazeer CSE-B*

#include<bits/stdc++.h>

#define ll long long

#define loop(*i*,*n*) for(int i = 0; i < n; i++)

#define loop1(*i*,*n*) for(int i = 1; i <= n; i++)

using namespace std;

int main()

{

    int n,i=0,j=0;

    cin>>n;

    int a[n];

    loop(i,n) {

        cin>>a[i];

    }

    for(; i < n; i++) {

        for(j = i+1; j < n; j++){

            if(a[i]<a[j]){

                a[i] = a[i] + a[j];

                a[j] = a[i] - a[j];

                a[i] = a[i] - a[j];

            }

        }

    }

    loop(i,n) {

        cout<<a[i]<<" ";

    }

    return 0;

}

**Screenshot:**

**A screen shot of a computer

Description automatically generated with low confidence**

1. **Find the factorial of a large number. A factorial of a number like 100 has 158 digits. It is not possible to store these many digits even if we use long int. So try using array to solve the problem.**

Input: 100

Output: 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000.

**Code:**

*//21161 Shaik Nazeer CSE-B*

#include<bits/stdc++.h>

#define ll long long

#define loop(*i*,*n*) for(int i = 0; i < n; i++)

#define loop1(*i*,*n*) for(int i = 1; i <= n; i++)

using namespace std;

void fact(int *i*,vector<int> &*ans*){

    int j = 0,carry=0,temp=0,len = *ans*.size();

    for(; j < len; j++){

        temp = *ans*[j]\**i* + carry;

*ans*[j] = temp%10;

        carry= temp/10;

    }

    while(carry){

*ans*.push\_back(carry%10);

        carry = temp%10;

    }

}

int main()

{

    int n,i;

    cin>>n;

    vector<int> ans;

    ans.push\_back(1);

    for(i = 2; i <=n; i++) {

        fact(i,ans);

    }

    reverse(ans.begin(),ans.end());

    int len = ans.size();

    for(i = 0; i < len; i++){

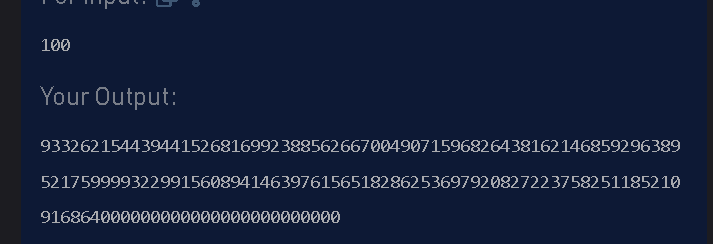
        cout<<ans[i];

    }

    return 0;

}

**Screenshot:**

****