**C++ Interns Questions (Tutree)**

**Paper 1**

**Find the smallest and second smallest elements in an array.**

**Input: arr[] = {12, 13, 1, 10, 34, 1}**

**Output: The smallest element is 1 and**

**second Smallest element is 10**

Code

#include<bits/stdc++.h>

using namespace std;

int main()

{

int n;

int arr[n];

cin>>n;

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

{

cin>>arr[i];

}

int min = INT\_MAX;

int second\_min = INT\_MAX ;

if (n < 2)

{

cout <<"\nInvalid Input";

}

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

{

if(arr[i] < min)

{

second\_min = min;

min = arr[i];

}

else if(arr[i] < second\_min && arr[i] != min)

{

second\_min = arr[i];

}

}

cout<<"The smallest element is " <<min<< " and second Smallest element is "<<second\_min<<endl;

return 0;

}

**Q2-Find median in a stream of integers (running integers)**

**Explanation-**

**Given that integers are read from a data stream. Find median of elements read so for in efficient way. For simplicity assume there are no duplicates. For example, let us consider the stream 5, 15, 1, 3 …**

#include<bits/stdc++.h>

using namespace std;

priority\_queue<int, vector<int>, greater <int> > min\_heap;

priority\_queue<int> max\_heap;

void add(int a)

{

if( max\_heap.size() && a >= max\_heap.top())

min\_heap.push(a);

else

max\_heap.push(a);

if(abs(max\_heap.size() - min\_heap.size()) > 1)

{

if(max\_heap.size() > min\_heap.size())

{

int temp = max\_heap.top();

max\_heap.pop();

min\_heap.push(temp);

}

else

{

int temp = min\_heap.top();

min\_heap.pop();

max\_heap.push(temp);

}

}

}

double get\_median()

{

int total = min\_heap.size() + max\_heap.size();

double ret;

if(total%2 == 1)

{

if(max\_heap.size() > min\_heap.size())

ret = max\_heap.top();

else

ret = min\_heap.top();

}

else

{

ret = 0;

if(max\_heap.empty() == false)

ret += max\_heap.top();

if(min\_heap.empty() == false)

ret += min\_heap.top();

ret/=2;

}

return ret;

}

int main()

{

cout << setprecision(1) << fixed;

int n, a;

cin >> n;

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

{

cin >> a;

add(a);

cout << get\_median() << endl;

}

}

**Paper - 2**

**Write a program to print all the LEADERS in the array. An element is leader if it is greater than all the elements to its right side. And the rightmost element is always a leader. For example int the array {16, 17, 4, 3, 5, 2}, leaders are 17, 5 and 2.**

Code

#include<bits/stdc++.h>

using namespace std;

int main()

{

// ios::sync\_with\_stdio(false);

string input;

int i,j,s,sum=0,k=0,target;

getline(cin,input);

j=input.size();

int arr[j];

for(i=0;i<j;i++)

{

if(input[i]!=' ')

{

s=int(input[i])-48;

//cout<<s<<endl;

sum=sum\*10+s;

}

else

{

arr[k++]=sum;

sum=0;

}

}

arr[k++]=sum;

int l=0,min=0;

j=k;

int arr2[j];

arr2[l++]=arr[j-1];

min = arr[j-1];

// count << min<<endl;

for(i=(j-2);i>=0;i--)

{

if(arr[i]>min)

{

arr2[l++]=arr[i];

min=arr[i];

}

}

for(i=l-1;i>=0;i--)

{

cout<<arr2[i]<<" ";

}

return 0;

}

**2. Program to print following pattern:**

**Input:- 5**

**\*                  \***

**\* \*              \* \***

**\* \* \*          \* \* \***

**\* \* \* \*      \* \* \* \***

**\* \* \* \* \*  \* \* \* \* \***

**\* \* \* \* \*  \* \* \* \* \***

**\* \* \* \*      \* \* \* \***

**\* \* \*          \* \* \***

**\* \*              \* \***

**\*                  \***

Code

#include<bits/stdc++.h>

using namespace std;

int main()

{

int T,x,i,j;

cin>>T;

x=T;

while((--x)>=0)

{

for(i=0;i<(T-x);i++)

{

cout<<"\* ";

}

for(j=T;j>(T-x);j--)

{

cout<<" ";

}

for(j=T;j>(T-x);j--)

{

cout<<" ";

}

for(i=0;i<(T-x);i++)

{

cout<<"\* ";

}

cout<<endl;

}

x=-1;

while((x++)<T)

{

for(i=0;i<(T-x);i++)

{

cout<<"\* ";

}

for(j=T;j>(T-x);j--)

{

cout<<" ";

}

//cout<<" ";

for(j=T;j>(T-x);j--)

{

cout<<" ";

}

for(i=0;i<(T-x);i++)

{

cout<<"\* ";

}

cout<<endl;

}

return 0;

}

**Paper – 3**

Q1. Union and Intersection of two sorted arrays.

#include<bits/stdc++.h>

using namespace std;

void printUnion(int arr1[], int arr2[], int len1, int len2) {

int f, i, j, k = 0;

int arr3[len1+len2];

for (i = 0; i < len1; i++) {

arr3[k] = arr1[i];

k++;

}

for (i = 0; i < len2; i++) {

f = 0;

for (j = 0; j < len1; j++) {

if (arr2[i] == arr1[j]) {

f = 1;

}

}

if (f == 0) {

arr3[k] = arr2[i];

k++;

}

}

cout << "Union of two array is:";

for (i = 0; i < k; i++) {

cout << arr3[i] << " ";

}

}

void printIntersection(int arr1[], int arr2[], int len1, int len2) {

int arr3[len1+len2];

int i, j, k = 0;

for (i = 0; i < len1; i++) {

for (j = 0; j < len2; j++) {

if (arr1[i] == arr2[j]) {

arr3[k] = arr1[i];

k++;

}

}

}

cout << "\nIntersection of two array is:";

for (i = 0; i < k; i++) {

cout << arr3[i] << " ";

}

}

int main()

{

int len1, len2;

cin>>len1;

int arr1[len1];

for (int i = 0; i < len1; i++)

{

cin >> arr1[i];

}

cin>>len2;

int arr2[len2];

for (int i = 0; i < len2; i++)

{

cin >> arr2[i];

}

printUnion(arr1, arr2, len1, len2);

printIntersection(arr1, arr2, len1, len2);

return 0;

}

Q2. Program for array rotation.

Write a function rotate(arr[], d, n) that rotates arr[] of size n by d elements.

Rotation of the above array by 2 will make array

Input: arr[] = {1,2,3,4,5,6}, n = 6, d = 2

Output: arr[] = {3,4,5,6,1,2}

#include<bits/stdc++.h>

using namespace std;

void Rotate(int arr[],int n,int d)

{

d=d%n;

for(int j=0;j<d;j++)

{

int temp = arr[0], i;

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

arr[i] = arr[i + 1];

arr[i] = temp;

}

}

int main()

{

int n,d;

cin>>n>>d;

int arr[n];

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

{

cin >> arr[i];

}

Rotate(arr,n,d);

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

{

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

}

cout<<endl;

return 0;

}

**Paper - 4**

**Q2. Find duplicates in O(n) time and O(1) extra space.**

Given an array of n elements that contains elements from 0 to n-1, with any of these numbers appearing any number of times. Find these repeating numbers in O(n) and using only constant memory space.

Input : n = 7 and array[] = {1, 2, 3, 6, 3, 6, 1}

Output: 1, 3, 6

Explanation: The numbers 1 , 3 and 6 appear more than once in the array

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin>>n;

int arr[n];

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

{

cin>>arr[i];

}

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

arr[arr[i] % n] = arr[arr[i] % n] + n;

}

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

if (arr[i] >= n \* 2) {

cout << i << " " << endl;

}

}

return 0;

}

**Q3. Find the minimum distance between two numbers.**

Given an unsorted array arr[] and two numbers x and y, find the minimum distance between x and y in arr[]. The array might also contain duplicates. You may assume that both x and y are different and present in arr[].

#include <bits/stdc++.h>

using namespace std;

int main()

{

int N;

cin>>N;

int X , Y;

int arr[N];

for(int i=0;i<N;i++)

{

cin>>arr[i];

}

cin>>X>>Y;

int min\_dist = INT\_MAX;

int i = 0, j = 0;

while(i < N and j < N)

{

if(arr[i] == X)

{

while( j < N and arr[j] != Y)

j++;

if(j < N and arr[j] == Y)

min\_dist = min(min\_dist,abs(i-j));

i = j;

}

else if(arr[i] == Y)

{

while( j < N and arr[j] != X)

j++;

if(j < N and arr[j] == X)

min\_dist = min(min\_dist,abs(i-j));

i = j;

}

else

i++;

}

cout<<min\_dist;

return 0;

}