# Even Odd Array

Accept n numbers from user in an array and Put even and odd elements of an array in two separate arrays and display them.

Input:

10

1 2 3 4 5 6 7 8 9 10

Output:

2 4 6 8 10

1 3 5 7 9

Constraints

0<n<=10

void copyContents(int a[],int even[],int odd[],int n)

{

int i,e=0,o=0;

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

if((a[i]%2)==0){

even[e]=a[i];

e++;

}

else{

odd[o]=a[i];

o++;

}

}

for(i=0;i<e;i++) printf("%d ",even[i]);

printf("\n");

for(i=0;i<o;i++) printf("%d ",odd[i]);

}

int main()

{

int arr[SIZE], i, n;

int even[SIZE], odd[SIZE], evenCount, oddCount;

scanf("%d", &n);//size of an array

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

{

scanf("%d", &arr[i]);//accept contents of array

}

copyContents(arr,even,odd,n);

return 0;

}

# Sum Of Elements

Given an Array, Accept n numbers from user and find if there exists a index whose sum of elements in left equals the sum of elements at right.

For example:

If the given array is 12346,

Index position '3' satisfies the condition. i.e.,1+2+3 = 6

If the given array is 5237,

position '2' satisfies the condition. 5+2 = 7

In case there is no Index that matches the condition display -1

Input:

1

5

12346

Output:

YES 3

Sample Input

1

7

1 0 0 2 1 0 0

Sample output

YES 3

int display(int arr[],int n )

{

int a,b,sr=0,sl=0,lock=0,i;

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

sr=0;sl=0;

for(a=0;a<i;a++) sl=sl+arr[a];

for(b=i+1;b<n;b++) sr=sr+arr[b];

if(sl==sr){lock=1;break;}

}

if(lock==0)return -1;

else return i;

}

int main(void)

{

int T,a[SIZE],i,index,n;

scanf("%d",&T);

while(T--)

{

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

index=display(a,n);

if(index==-1)

printf("NO\n");

else

printf("YES %d\n",index);

}

return 0;

}

# Frequency Counter

Given an array, accept n numbers and display the frequency of the elements

Input

5

1 2 3 1 2

Output

1 2

2 2

3 1

Constraints 0<n<=10

void countFreq(int arr[],int freq[],int n)

{

int a[1000]={0},an=0,i,j,l,f=0,c;

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

l=1;c=0;

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

if(arr[i]==a[j])l=l-1000;

else l=l+1;

}

if(l>0){

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

if(arr[i]==arr[j]){c++;}

}

a[an]=arr[i];an++;

freq[f]=c;

f++;

}

else {freq[f]=0;

f++;}

}

}

int main()

{

int arr[SIZE], freq[SIZE];

int n, i, j, count;

scanf("%d", &n);

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

{

scanf("%d", &arr[i]);

freq[i] = -1;

}

countFreq(arr,freq,n);

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

{

if(freq[i]!=0)

{

printf("%d %d\n",arr[i], freq[i]);

}

}

return 0;

}

# Sum of Prime Numbers

Write a program to accept a number n from user and accept n elements in an array

Display Prime numbers, count of prime numbers and the Sum of all Prime numbers entered in an array

INPUT

5

11 33 44 66 13

OUTPUT

11 13 2 24

Explanation 11 and 13 are prime numbers 2 is count of prime numbers and 24 is sum of prime numbers

void displayCount(int a[],int n)

{

int e,i,b,flag,c=0,sum=0;

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

e=a[i];flag=0;b=2;

while(b<e){

if(e%b==0){flag=1;

break;}

b++;

}

if(flag==0){printf("%d ",e);c++;sum=sum+e;}

}

printf("%d %d",c,sum);

}

int main()

{

int a[SIZE],count,c,n,i,j,sum=0;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

displayCount(a,n);

return 0;

}

# Leaders

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.

Let the input array be arr[] and size of the array be N.

constraints

1<=N<10000

1<=arr[i]<10000

Sample Input

6

16 17 4 3 5 2

Sample output

17 5 2

Explanation: In sample input 6 represent the size of the array and second line indicates elements of the array.

In Sample input 17 is greater than all numbers on its right side ,5 is also greater than all its right numbers and 2 is rightmost element.

void printLeaders(int arr[], int n)

{int i,j,lock,e;

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

e=arr[i];lock=0;

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

if(e>arr[j]&&i<=j)lock=lock+1;

else lock=lock-1000;

}

if(lock>0) printf("%d ",e);

}

printf("%d",arr[n-1]);

}

int main()

{

int a[SIZE],i,j,n;

scanf("%d",&n);

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

scanf("%d",&a[i]);

printLeaders(a,n);

return 0;

}

# Maximum Triplet Product

Given an integer array arr[N] having N elements, find a maximum product of a triplet in array.

Constraints

1<=N<1000

-1<arr[i]<10000

Sample input

5

10 3 5 6 20

Sample output

1200

Explanation:In Sample input 5 indicates size of array and second line indicates N elements in the array.

Sample output is displaying maximum triplet product i.e(10X6X20 = 1200)

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <assert.h>

#include <limits.h>

#include <stdbool.h>

int main() {

int array[100], n, c, d, position, swap,f=1;

scanf("%d", &n);

for ( c = 0 ; c < n ; c++ )

scanf("%d", &array[c]);

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

{

position = c;

for ( d = c + 1 ; d < n ; d++ )

{

if ( array[position] > array[d] )

position = d;

}

if ( position != c )

{

swap = array[c];

array[c] = array[position];

array[position] = swap;

}

}

for(c=n-1;c>=n-3;c--){

f=f\*array[c];

}

printf("%d",f);

return 0;

}

# Selection Sort

U have been given the task to accept n elements in array and display the the elements after every pass

Sample Input

7

64 34 25 12 22 11 90

Sample Output

11 34 25 12 22 64 90

11 12 25 34 22 64 90

11 12 22 34 25 64 90

11 12 22 25 34 64 90

11 12 22 25 34 64 90

11 12 22 25 34 64 90

11 12 22 25 34 64 90

Explanation

7 is number of elements

64 34 25 12 22 11 90 are elements of an array

11 34 25 12 22 64 90 is contents of array after first pass

11 12 25 34 22 64 90 after second pass

11 12 22 34 25 64 90 after third pass

11 12 22 25 34 64 90 after fourth pass

and so on....

void selectionSort(int a[], int n)

{int i,j,k,id=0,small,temp,lock;

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

small=a[i];lock=0;

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

if(small>a[j]){small=a[j];id=j;lock=1;}

}if(lock==1){

temp=a[id];

a[id]=a[i];

a[i]=temp;}

for(k=0;k<n;k++)printf("%d ",a[k]);

printf("\n");

}

}

int main()

{

int a[100];

int i,n;

scanf("%d",&n);

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

scanf("%d",&a[i]);

selectionSort(a, n);

return 0;

}

# Minimum Number Of Swaps

Problem statement

As the problem title suggests, you need to count the number of swaps required to sort a given list of integers in ascending order using the selection sort algorithm.

Input

First line of the input contains the number of test cases T. T test cases follow. The first line of each test case contains the N, (1 ≤ N ≤ 50) the number of integers in the array. The second line contains N space separated integers. All N numbers are distinct.

Output

For each test case, print on one line, the number of swaps (minimum) to sort the given array in ascending order using selection sort.

Sample Input

2

3

4 2 5

5

10 11 8 7 1

Sample Output

1

3

void selectionSort(int array[], int n)

{

count=0;

int c, d, position, swap,lock;

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

{

position = c;lock=1;

for ( d = c + 1 ; d < n ; d++ )

{

if ( array[position] > array[d] )

position = d;

}

if ( position != c )

{

swap = array[c];

array[c] = array[position];

array[position] = swap;

count++;

}

}

}