1

#include<iostream>

using namespace std;

int Merge(int A[], int low, int mid, int high)

{

int C[30];

int index = low;

int v = 0;

int i = low;

int j = (mid + 1);

while(i<=mid && j<=high)

{

if(A[i]<=A[j])

{

C[index]=A[i];

i++;

}

else

{

C[index]=A[j];

j++;

v+=(mid-i+1);

}

index++;

}

if(i>mid)

{

for(int p=j; p<=high; p++)

{

C[index]=A[p];

index++;

}

}

else

{

for(int q=j; q<=mid; q++)

{

C[index]=A[q];

index++;

}

}

for(int k=low; k<=high; k++)

{

A[k]=C[k];

}

return v;

}

int icount(int A[], int low, int high)

{

int mid, c1, c2, c3;

if(low<high)

{

mid = (low + high)/2;

c1 = icount(A, low, mid);

c2 = icount(A, mid+1, high);

c3 = Merge(A, low, mid, high);

return(c1+c2+c3);

}

else

return 0;

}

int main()

{

int A[40];

int n, i;

cout<<"\nEnter size of array : ";

cin>>n;

cout<<"\nEnter elements of array : "<<endl;

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

{

cin>>A[i];

}

cout<<"\nNumber of inversions : "<<icount(A, 0, n-1);

}

2

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

unsigned int numberOfTimesDivisibleBy5(unsigned int n)

{

return (n % 5 == 0) ? 1 + numberOfTimesDivisibleBy5(n / 5) : 0;

}

int minimumNumber(int present, int sum, int value)

{

if (sum >= value)

return present - 1;

else

{

sum += numberOfTimesDivisibleBy5(present);

return minimumNumber(present + 1, sum, value);

}

}

int main()

{

int val;

cout << "\nEnter the value: ";

cin >> val;

cout << "Minimum number is " <<minimumNumber(1, 0, val);

}

3

#include <iostream>

using namespace std;

void Merge(int arr[], int l, int m, int r)

{

int n1 = m - l + 1;

int n2 = r - m;

// Create temp arrays

int L[n1], R[n2];

// Copy data to temp arrays L[] and R[]

for(int i = 0; i < n1; i++)

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

for(int j = 0; j < n2; j++)

R[j] = arr[m + 1 + j];

// Merge the temp arrays back into arr[l..r]

// Initial index of first subarray

int i = 0;

// Initial index of second subarray

int j = 0;

// Initial index of merged subarray

int k = l;

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

// Copy the remaining elements of L[], if there are any

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

// Copy the remaining elements of R[], if there are any

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

void Merge\_sort(int arr[], int l, int r)

{

if (l < r)

{

int m = (l+r) / 2;

// Sort first and second halves

Merge\_sort(arr, l, m);

Merge\_sort(arr, m + 1, r);

Merge(arr, l, m, r);

}

}

int main()

{

int A[30],n,k,i,specialty=0;

cout<<"\nEnter the size of Array : ";

cin>>n;

cout<<"\nEnter the Value of K : ";

cin>>k;

cout<<"\nEnter elements of the Array : "<< endl;

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

{

cin>>A[i];

}

Merge\_sort(A,0,n-1);

/\* HERE CONTIGUOUS SUBARRAY IS INTERPRETED AS SUBARRAY WHOSE ELEMENTS SHARE COMMON BORDER\*/

cout<<"\nArray after Merging based on specialty condition : ";

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

{

cout<<"\n"<<A[i];

}

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

{

specialty+=A[i];

}

cout<<"\nSpecialty of the sequence is : "<<specialty;

}

4

#include <iostream>

using namespace std;

int binarySearch(int arr[], int l, int r, int val)

{

if (r >= l) {

int mid = l + (r - l) / 2;

if (arr[mid] == val)

return mid;

if (arr[mid] > val)

return binarySearch(arr, l, mid - 1, val);

return binarySearch(arr, mid + 1, r, val);

}

return -1;

}

void merge(int arr[], int l, int m, int r)

{

int n1 = m - l + 1;

int n2 = r - m;

int L[n1], R[n2];

for(int i = 0; i < n1; i++)

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

for(int j = 0; j < n2; j++)

R[j] = arr[m + 1 + j];

int i = 0;

int j = 0;

int k = l;

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int arr[], int l, int r)

{

if (l < r)

{

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

int Minimize(int arr[], int size, int x)

{

int available = binarySearch(arr, 0, size, x+1);

if(available == -1)

return x+1;

while(available != -1)

available = binarySearch(arr, 0, size, ++x);

return x;

}

int main()

{

int s,i,x;

cout << "\nEnter the size of the Array: ";

cin >> s;

int A[s];

cout << "\nEnter the values of the array" << endl;

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

cin >> A[i];

cout << "\nEnter the value of X: " << endl;

cin >> x;

mergeSort(A, 0, s-1);

cout << "Minimized value of z is :" <<Minimize(A, s, x);;

}

5

#include<bits/stdc++.h>

#include<cmath>

using namespace std;

int partition(int arr[],int start, int end){

int pivot = arr[start];

int i = start;

int j = end;

while(i < j){

while(i < j && arr[--j] >= pivot);

//found the first element less than or equal to pivot

if(i < j){

arr[i] = arr[j];

}

while(i < j && arr[++i] <= pivot);

//found the element greater than or equal to pivot

if(i < j){

arr[j] = arr[i];

}

}

arr[j] = pivot;

return j;

}

void quickSort(int arr[],int start,int end){

if(end - start < 2) return;

int pivotIndex = partition(arr,start,end);

quickSort(arr,start,pivotIndex);

quickSort(arr,pivotIndex + 1,end);

}

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

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

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

}

cout << endl;

}

int main()

{

int n,k,t;

cout << "Enter the total number of test cases " << endl;

cin >> t;

while(t--)

{

cout << "Enter the values of N and K " << endl;

cin >> n >> k;

int arr[n];

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

{

cin >> arr[i];

}

quickSort(arr,0,n-1);

double sum = 0.0;

sum = (arr[n-1]-arr[1]+arr[n-2]-arr[0])\*2.0;

sum = sum + sqrt(2.0)\*(arr[n-1]-arr[n-2]+arr[1]-arr[0]);

long long rounded=ceil(sum);

cout << "The minimum cost is " << endl;

cout << rounded \* k << endl;

}

return 0;

}

1

// C++ implementation of the above approacdh

#include <bits/stdc++.h>

#define ll long long int

using namespace std;

// Function to calculate gcd of two number

ll gcd(ll a, ll b)

{

if (b == 0)

return a;

else

return gcd(b, a % b);

}

// Function to check if two numbers are coprime or not

bool coPrime(ll n1, ll n2)

{

// two numbers are coprime if their gcd is 1

if (gcd(n1, n2) == 1)

return true;

else

return false;

}

// Function to find largest integer less

// than or equal to N/2 and coprime with N

ll largestCoprime(ll N)

{

ll half = floor(N / 2);

// Check one by one all numbers

// less than or equal to N/2

while (coPrime(N, half) == false)

half--;

return half;

}

// Driver code

int main()

{

int n;

cin>>n;

cout << largestCoprime(n);

return 0;

}

2

#include <bits/stdc++.h>

using namespace std;

int main(){

int N, M;

cout << "Enter the values of N and M " << endl;

cin >> N >> M;

int arr[N][M] = { 1,0,1,0,

0,1,0,0,

0,0,1,0,

0,0,1,1};

/\* for(int i = 0; i < N; i++){

for(int j = 0; j < M; j++){

cin >> arr[i][j];

}

}\*/

int x = 0;

for(int i = 1; i < N; i++){

for(int j = 0; j < M; j++){

if(arr[i][j] > arr[x][j]){

x = i;

break;

}

else if(arr[i][j] < arr[x][j]) break;

}

}

cout << x + 1 << endl;

return 0;

}

3

#include <bits/stdc++.h>

using namespace std;

int main(){

string s,t;

cin >> s >> t;

vector<pair<int,char>> v;

for(int i = 0; i < t.length(); i++){

for(int j = 0; j < 26; j++){

if(t[i] == s[j]){

v.push\_back(make\_pair(j,s[j]));

}

}

}

sort(v.begin(),v.end());

for(int i = 0; i < t.length(); i++){

cout << v[i].second;

}

return 0;

}