**Problem 1 : Implement Insertion Sort**

#include<bits/stdc++.h>

#define ll long long int

using namespace std;

void InsertionSort(ll arr[],ll n)

{

ll i,j,key;

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

{

key = arr[i];

j=i-1;

while(j>=0 && arr[j]>key)

{

//swap arr[j] and arr[j+1]

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

arr[j] = key;

j--;

}

}

}

int main()

{

ll n,i;

cout<<"Enter the number of elements : ";

cin>>n;

ll a[n];

cout<<"\nEnter the Elements : ";

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

cin>>a[i];

InsertionSort(a,n);

cout<<"\nThe Sorted array is :";

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

cout<<a[i]<<" ";

return 0;

}

**Problem 2 : Implement Merge Sort**

#include<bits/stdc++.h>

#define fo(i,a,b) for(i=a;i<b;i++)

#define ll long long int

using namespace std;

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

{

ll i,j,k;

ll n1=m-l+1;

ll n2=r-m;

ll left[n1], right[n2];

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

{

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

}

for(i=0;i<n2;i++)

{

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

}

i=0;

j=0;

k=l;

while(i<n1 && j<n2)

{

if(left[i]<=right[j])

{

arr[k]=left[i];

i++;

}

else

{

arr[k]=right[j];

j++;

}

k++;

}

while(i<n1)

{

arr[k]=left[i];

i++;

k++;

}

while(j<n2)

{

arr[k]=right[j];

j++;

k++;

}

}

void MergeSort(ll arr[], ll l, ll r)

{

if(l<r)

{

ll m;

m = l + (r-l)/2; //same as (l+r)/2 but avoids overflow

MergeSort(arr,l,m);

MergeSort(arr,m+1,r);

Merge(arr,l,m,r);

}

}

void Printarray(ll arr[],ll n)

{

ll i;

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

{

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

}

}

int main()

{

ll n,i;

cout << "Enter the size of Array : ";

cin >> n;

ll arr[n];

cout << "Enter the array elements : ";

fo(i,0,n)

cin >> arr[i];

MergeSort(arr,0,n-1);

Printarray(arr,n);

return 0;

}

**Problem 3 : Implement Bubble Sort**

#include<bits/stdc++.h>

#define ll long long int

using namespace std;

void BubbleSort(ll arr[], ll n)

{

ll i, j, flag, temp;

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

{

flag = 0;

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

{

if (arr[j] > arr[j + 1])

{

//swap the two numbers

flag = 1;

temp = arr[j];

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

arr[j + 1] = temp;

}

}

if (flag == 0)

break;

}

}

int main()

{

ll n, i;

cout << "Enter the number of elements : ";

cin >> n;

ll a[n];

cout << "\nEnter the Elements : ";

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

cin >> a[i];

BubbleSort(a, n);

cout << "\nThe Sorted array is :";

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

cout << a[i] << " ";

return 0;

}

**Problem 4 : Find Maximum Sum Subarray**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

int main()

{

ll n;

cout << "Enter size of array : ";

cin >> n;

ll arr[n];

ll i;

cout << "Enter array elements : ";

fo(i,0,n)

cin >> arr[i];

cout << "Max Sum of Subarray is : ";

ll sum[n];

sum[0] = arr[0];

fo(i,1,n)

{

if(sum[i-1] < 0)

sum[i] = arr[i];

else

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

}

sort(sum,sum+n);

cout << sum[n-1];

return 0;

}

**Problem 5 : Implement Heap Sort**

#include<bits/stdc++.h>

#define ll long long int

using namespace std;

void heapify(ll a[], ll n, ll i)

{

ll lt = i;

ll lc = 2 \* i + 1;

ll rc = 2 \* i + 2;

if (lc < n && a[lc] > a[lt])

lt = lc;

if (rc < n && a[rc] > a[lt])

lt = rc;

if (lt != i)

{

swap(a[i], a[lt]);

heapify(a, n, lt);

}

}

void heapsort(ll a[], ll n)

{

for (ll i = n / 2 - 1; i >= 0; i--)

heapify(a, n, i);

for (ll i = n - 1; i >= 0; i--)

{

swap(a[0], a[i]);

heapify(a, i, 0);

}

}

int main()

{

ll n;

cout << "Enter the size of the array : ";

cin >> n;

ll a[n];

cout << "Enter the elements of the array : " << '\n';

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

cin >> a[i];

cout << "Array after sorting :" << '\n';

heapsort(a, n);

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

cout << a[i] << " ";

}

**Problem 6 : Implement Radix Sort**

#include<bits/stdc++.h>

#define ll long long int

using namespace std;

ll getmax(ll a[], ll n)

{

ll max = a[0];

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

{

if (max < a[i])

max = a[i];

}

return max;

}

void countsort(ll a[], ll n, ll pos)

{

ll output[n];

ll i, count[10] = {0};

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

count[(a[i] / pos) % 10]++;

for (ll i = 1; i < 10; i++)

count[i] = count[i] + count[i - 1];

for (ll i = n - 1; i >= 0; i--)

output[--count[(a[i] / pos) % 10]] = a[i];

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

a[i] = output[i];

}

void radixsort(ll a[], ll n)

{

ll max = getmax(a, n);

for (ll i = 1; max / i > 0; i = i \* 10)

countsort(a, n, i);

}

int main()

{

cout << "Enter the size of the array : ";

ll n;

cin >> n;

cout << "Enter the elements of the array : " << '\n';

ll a[n];

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

cin >> a[i];

cout << "Array after radix sort is : " << '\n';

radixsort(a, n);

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

{

cout << a[i] << " ";

}

}

**Problem 7 : Implement Bucket Sort**

#include<bits/stdc++.h>

#define ll long long int

using namespace std;

void bucketsort(double a[], ll n)

{

vector<double>v[n];

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

{

ll b = n \* a[i];

v[b].push\_back(a[i]);

}

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

{

sort(v[i].begin(), v[i].end());

}

ll index = 0;

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

{

for (ll j = 0; j < v[i].size(); j++)

{

a[index++] = v[i][j];

}

}

}

int main()

{

ll n;

cout << "Enter the size of the Array : ";

cin >> n;

double a[n];

cout << "Enter the elements of the array : " << '\n';

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

{

cin >> a[i];

}

cout << "Array after sorting :" << '\n';

bucketsort(a, n);

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

{

cout << a[i] << " ";

}

cout << '\n';

}

**Problem 8 : Program to find nth Fibonacci Sequence using 3 Methods**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

void Multiply(ll A[2][2], ll B[2][2])

{

ll mul[2][2],i,j,k;

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

mul[i][j] = 0;

for (k = 0; k < 2; k++)

mul[i][j] += A[i][k] \* B[k][j];

}

}

for (i=0; i<2; i++)

for (j=0; j<2; j++)

A[i][j] = mul[i][j];

}

void FastPow(ll F[2][2], ll n)

{

if(n == 0 || n == 1)

return;

ll M[2][2] = {{1,1},{1,0}};

FastPow(F, n/2);

Multiply(F, F);

if (n%2 != 0)

Multiply(F, M);

}

ll FiboM(ll n)

{

ll A[2][2] = {{1,1},{1,0}};

if (n == 0)

return 0;

FastPow(A, n-1);

return A[0][0];

}

ll RecFibo(ll n)

{

if(n<=1)

return n;

else

return RecFibo(n-1) + RecFibo(n-2);

}

ll FiboDP(ll n)

{

ll i,arr[n+1];

arr[1] = 1;

arr[2] = 1;

foe(i,3,n)

{

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

}

return arr[n];

}

int main()

{

ll i,n;

cout << "Enter value on n : ";

cin >> n;

cout << "Matrix Exponentiation [nth Fibo term] : " << FiboM(n) << '\n';

cout << "Recursive Function [nth Fibo term] : " << RecFibo(n) << '\n';

cout << "DP array [nth Fibo Term] : " << FiboDP(n) << '\n';

}

**Problem 9 : Implement Binary Search**

#include<bits/stdc++.h>

#define ll long long int

using namespace std;

int BinarySearch(ll arr[], ll low, ll high, ll key)

{

ll mid;

while (low <= high)

{

mid = (low + high) / 2;

if (arr[mid] > key)

high = mid - 1;

else if (arr[mid] < key)

low = mid + 1;

else

return mid;

}

return -1;

}

int main()

{

ll n, i, temp;

cout << "Enter the Size of the array : ";

cin >> n;

ll arr[n];

cout << "\nEnter the elements of the array in ascending order : ";

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

cin >> arr[i];

cout << "\nEnter the element you want to search in the array : ";

cin >> temp;

temp = BinarySearch(arr, 0, n - 1, temp);

if (temp == -1)

cout << "\nThe element was not found";

else

cout << "\nThe element was found at index " << temp;

}

**Problem 10 : Write a program to find local minima in an array**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

ll Check(ll arr[],ll n,ll c)

{

ll right,left,ele;

ele = arr[c];

right = arr[(c+1 >= n ? c : c+1)];

left = arr[(c-1 < 0 ? c : c-1)];

if(right < ele)

return 1;

else if(left < ele)

return -1;

else

return 0;

}

ll LocalMinima(ll arr[],ll n,ll low, ll high)

{

ll check,mid;

while(low <= high)

{

mid = low + (high-low)/2;

check = Check(arr,n,mid);

if(check == 0)

return arr[mid];

else if(check == 1)

low = mid + 1;

else

high = mid - 1;

}

}

int main()

{

ll n;

cout << "Enter size of array : ";

cin >> n;

ll arr[n],i;

cout << "Enter elements of array : ";

fo(i,0,n)

{

cin >> arr[i];

}

cout << "Local Minima in Array : " << LocalMinima(arr,n,0,n-1);

return 0;

}

**Problem 11 : Write a program to find local minima in a grid**

#include<bits/stdc++.h>

#define boost ios::sync\_with\_stdio(false); cin.tie(0)

#define ll long long int

#define mk make\_pair

#define pb push\_back

#define f first

#define s second

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

const long long int N = 1000;

ll arr[N][N];

ll FindMin(ll n,ll c)

{

ll i,min\_ind = 0;

fo(i,0,n)

if(arr[i][c] < arr[min\_ind][c])

min\_ind = i;

return min\_ind;

}

ll Check(ll n,ll r,ll c)

{

ll right,left,ele;

ele = arr[r][c];

right = arr[r][(c+1 >= n ? c : c+1)];

left = arr[r][(c-1 < 0 ? c : c-1)];

if(right < ele)

return 1;

else if(left < ele)

return -1;

else

return 0;

}

ll LocalMinimaCC(ll n,ll low, ll high)

{

ll min,mid,check;

ll cnt = 1;

while(low <= high)

{

mid = (low+high)/2;

min = FindMin(n,mid);

check = Check(n,min,mid);

if(check == 0)

return arr[min][mid];

else if(check == 1)

low = mid + 1;

else

high = mid - 1;

cnt++;

}

}

int main()

{

ll i,j,n;

cout << "Enter dimension [n] of (n\*n) Grid : ";

cin >> n;

cout << "Enter Grid Elements : \n";

fo(i,0,n)

fo(j,0,n)

cin >> arr[i][j];

cout << "\nLocal Minima in Grid is : " << LocalMinimaCC(n,0,n-1);

return 0;

}

**Problem 12 : Implement Matrix Multiplication**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

int main()

{

ll m,n,y,z,i,j,k;

cout << "Enter dimensions of Matrix 1 : ";

cin >> m >> n;

cout << "Enter the dimensions of Matrix 2 : ";

cin >> y >> z;

if(n==y)

{

ll arr1[m][n],arr2[y][z];

cout << "Enter elements of Matrix 1: ";

fo(i,0,m)

fo(j,0,n)

cin >> arr1[i][j];

cout << "Enter elements of Matrix 2: ";

fo(i,0,y)

fo(j,0,z)

cin >> arr2[i][j];

ll mul[m][z];

fo(i,0,m)

{

fo(j,0,z)

{

mul[i][j] = 0;

fo(k,0,n)

mul[i][j] += arr1[i][k]\*arr2[k][j];

}

}

cout << "Matrix 1 \* Matrix 2 : \n";

fo(i,0,m)

{

fo(j,0,z)

cout << mul[i][j] << ' ';

cout << '\n';

}

}

else

cout << "\nCant Multiply the given two matrices : ";

}

**Problem 13 : Write a program to find GCD and LCM of 2 numbers**

#include<bits/stdc++.h>

#define ll long long int

using namespace std;

ll Gcd(ll a, ll b)

{

if(b==0)

return a;

else

return Gcd(b,a%b);

}

int main()

{

ll a,b;

cout << "Enter two numbers : ";

cin >> a >> b;

ll gcd = Gcd(max(a,b),min(a,b));

cout << "GCD : " << gcd << '\n';

cout << "LCM : " << (a\*b)/gcd;

return 0;

}

**Problem 14 : Implement Range Minima Problem**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

const int N = 10;

ll pre[10000][N];

ll BruteRangeMinima(ll arr[], ll n,ll l, ll r)

{

ll i,min;

min = l;

foe(i,l,r)

{

if(arr[i] < arr[min])

min = i;

}

return arr[min];

}

ll HP2(ll n)

{

ll p = (ll)log2(n);

return (ll)pow(2, p);

}

ll RangeMinima(ll i, ll j)

{

ll len;

len = j-i+1;

if(len == HP2(len))

{

ll temp;

temp = (ll)log2(len);

return pre[i][temp];

}

else

{

ll g,h;

g = HP2(len);

h = (ll)log2(len);

return min(pre[i][h], RangeMinima(i+g,j));

}

}

int main()

{

ll n;

cout << "Enter the size of the array : ";

cin >> n;

ll arr[n+1];

ll i,j,k;

cout << "Enter the array elements : \n";

foe(i,1,n)

{

cin >> arr[i];

}

foe(i,1,n)

{

bool over = false;

foe(j,0,N)

{

ll min = i;

foe(k,i,i+pow(2,j)-1)

{

if(k>n)

{

over = true;

break;

}

if(arr[k] < arr[min])

min = k;

}

pre[i][j] = arr[min];

if(over)

break;

}

}

ll q;

cout << "Enter number of queries : ";

cin >> q;

fo(i,0,q)

{

ll l,r;

cout << "\nEnter Query " << i+1 << " :";

cin >> l >> r;

cout << "Range Minimum [Brute] : "<< BruteRangeMinima(arr,n,l,r);

cout << "\nRange Minimum [Optimal] : " << RangeMinima(l,r);

}

return 0;

}

**Problem 15 : Implement Binary Tree and Binary Search Tree**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

struct node

{

int key;

struct node \*left, \*right;

};

struct node \*newNode(int item)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->key = item;

temp->left = temp->right = NULL;

return temp;

}

void inorder(struct node \*root)

{

if (root != NULL)

{

inorder(root->left);

printf("%d ", root->key);

inorder(root->right);

}

}

struct node\* insert(struct node\* node, int key)

{

if (node == NULL) return newNode(key);

if (key < node->key)

node->left = insert(node->left, key);

else if (key > node->key)

node->right = insert(node->right, key);

return node;

}

struct node\* search(struct node\* root, int key)

{

if (root == NULL || root->key == key)

return root;

if (root->key < key)

return search(root->right, key);

return search(root->left, key);

}

struct node \* minValueNode(struct node\* node)

{

struct node\* current = node;

while (current && current->left != NULL)

current = current->left;

return current;

}

struct node\* deleteNode(struct node\* root, int key)

{

if (root == NULL) return root;

if (key < root->key)

root->left = deleteNode(root->left, key);

else if (key > root->key)

root->right = deleteNode(root->right, key);

else

{

if (root->left == NULL)

{

struct node \*temp = root->right;

free(root);

return temp;

}

else if (root->right == NULL)

{

struct node \*temp = root->left;

free(root);

return temp;

}

struct node\* temp = minValueNode(root->right);

root->key = temp->key;

root->right = deleteNode(root->right, temp->key);

}

return root;

}

int main()

{

struct node \*root = NULL;

root = insert(root, 50);

insert(root, 30);

insert(root, 20);

insert(root, 40);

insert(root, 70);

insert(root, 60);

insert(root, 80);

inorder(root);

int key;

printf("\nEnter Element you wish to search : ");

scanf("%d",&key);

if(search(root,key))

{

printf("\nElement found in the tree!");

}

else

{

printf("\nElement not found in the tree");

}

cout << "\nEnter node you wish to delete : ";

cin >> key;

if(search(root,key))

root = deleteNode(root, key);

cout << '\n';

inorder(root);

return 0;

}

**Problem 16 : Implement Shortest Path in a grid**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

int main()

{

ll m,i,n,u;

cout << "Enter Dimensions of grid : ";

cin >> m >> n;

bool arr[(m\*n)];

fo(i,0,m\*n)

arr[i] = true;

cout << "Enter the number of Blocked cells : ";

ll temp;

cin >> temp;

cout << "Enter the positions of Blocked cells (x,y) : \n";

while(temp--)

{

ll x,y;

cin >> x >> y;

arr[(x\*m)+y] = false;

}

ll source,sx,sy,dx,dest,dy;

cout << "Enter Source Coordinates : ";

cin >> sx >> sy;

source = (sx\*m) + sy;

cout << "Enter Destination Coordinates : ";

cin >> dx >> dy;

dest = (dx\*m) + dy;

queue <ll> q;

ll dist[m\*n];

fo(i,0,m\*n)

dist[i] = -1;

q.push(source);

dist[source] = 0;

while(!q.empty())

{

temp = q.front();

q.pop();

if(temp - n < 0 ? false : true) //Element above

{

u = temp - n;

if(arr[u]) //If unblocked

{

if(dist[u] == -1) //If unvisited

{

dist[u] = dist[temp] + 1;

q.push(u);

}

}

}

if(temp + n >= (m\*n) ? false : true) //Element below

{

u = temp + n;

if(arr[u])

{

if(dist[u] == -1)

{

dist[u] = dist[temp] + 1;

q.push(u);

}

}

}

if(temp % n == 0 ? false : true) //Element to the left

{

u = temp - 1;

if(arr[u])

{

if(dist[u] == -1)

{

dist[u] = dist[temp] + 1;

q.push(u);

}

}

}

if((temp+1) % n == 0 ? false : true) //Element to the right

{

u = temp + 1;

if(arr[u])

{

if(dist[u] == -1)

{

dist[u] = dist[temp] + 1;

q.push(u);

}

}

}

}

if(dist[dest] == -1) //Destination not visited

{

cout << "No Path available from source to destination!";

}

else

cout << "Shortest path from Source to dest is : " << dist[dest];

return 0;

}

**Problem 17 : Implement Majority Element Problem**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

int main()

{

ll n, i, cnt = 1;

cout << "Enter size of array : ";

cin >> n;

char arr[n];

cout << "Enter n characters : ";

fo(i, 0, n)

cin >> arr[i];

char ind = 0;

fo(i, 1, n)

{

if (arr[i] == arr[ind])

cnt++;

else

cnt--;

if (cnt == 0)

{

ind = i;

cnt = 1;

}

}

cnt = 0;

fo(i, 0, n)

{

if (arr[i] == arr[ind])

cnt++;

}

if (cnt > n / 2)

cout << "Majority element is : '" << arr[ind] << '\'';

else

cout << "No majority element in the array!";

}

**Problem 18 : Implement Karatsubas Algorithm to multiply two numbers**

#include<bits/stdc++.h>

#define ll long long int

using namespace std;

ll makeEqualLength(string &str1, string &str2)

{

ll i;

ll len1 = str1.size();

ll len2 = str2.size();

if (len1 < len2)

{

for (i = 0 ; i < len2 - len1 ; i++)

str1 = '0' + str1;

return len2;

}

else if (len1 > len2)

{

for (i = 0 ; i < len1 - len2 ; i++)

str2 = '0' + str2;

}

return len1;

}

string addBitStrings( string first, string second )

{

string result;

ll length = makeEqualLength(first, second);

ll carry = 0;

for (ll i = length-1 ; i >= 0 ; i--)

{

ll firstBit = first.at(i) - '0';

ll secondBit = second.at(i) - '0';

ll sum = (firstBit ^ secondBit ^ carry)+'0';

result = (char)sum + result;

carry = (firstBit&secondBit) | (secondBit&carry) | (firstBit&carry);

}

if (carry)

result = '1' + result;

return result;

}

// Multiply single bits of strings a and b

ll MultiplyiSingleBit(string a, string b)

{

return (a[0] - '0')\*(b[0] - '0');

}

ll Multiply(string X, string Y)

{

ll n;

n = makeEqualLength(X, Y);

// Base cases

if (n == 0)

return 0;

if (n == 1)

return MultiplyiSingleBit(X, Y);

ll fh = n/2;

ll sh = (n-fh);

string Xl = X.substr(0, fh);

string Xr = X.substr(fh, sh);

string Yl = Y.substr(0, fh);

string Yr = Y.substr(fh, sh);

ll P1 = Multiply(Xl, Yl);

ll P2 = Multiply(Xr, Yr);

ll P3 = Multiply(addBitStrings(Xl, Xr), addBitStrings(Yl, Yr));

return P1\*(1<<(2\*sh)) + (P3 - P1 - P2)\*(1<<sh) + P2;

}

int main()

{

// boost;

string num1, num2;

cout << "Enter two Binary Strings : ";

cin >> num1 >> num2;

cout << "\nMultiplication of two numbers is: " << Multiply(num1,num2);

return 0;

}

**Problem 19 : Write a program to Count inversions in an array**

#include<bits/stdc++.h>

#define fo(i,a,b) for(i=a;i<b;i++)

using namespace std;

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

{

int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

int left[n1], right[n2];

int inv\_count = 0; //newly added

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

{

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

}

for (i = 0; i < n2; i++)

{

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

}

i = 0;

j = 0;

k = l;

while (i < n1 && j < n2)

{

if (left[i] <= right[j])

{

arr[k] = left[i];

i++;

}

else

{

arr[k] = right[j];

j++;

inv\_count += (m + 1) - (l + i); //newly added

}

k++;

}

while (i < n1)

{

arr[k] = left[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = right[j];

j++;

k++;

}

return inv\_count; //newly added

}

int MergeSort(int arr[], int l, int r)

{

int inv\_count = 0; //newly added

if (l < r)

{

int m;

m = l + (r - l) / 2; //same as (l+r)/2 but avoids overflow

inv\_count = MergeSort(arr, l, m);

inv\_count += MergeSort(arr, m + 1, r);

inv\_count += Merge(arr, l, m, r);

}

return inv\_count;

}

void Printarray(int arr[], int n)

{

int i;

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

{

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

}

}

int main()

{

int n;

cout << "Enter size of the array : ";

cin >> n;

int arr[n];

cout << "Enter array elements : ";

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

cin >> arr[i];

cout << "Number of inversions : " << MergeSort(arr, 0, n - 1) << "\n";

return 0;

}

**Problem 20 : Implement Quick Sort**

#include<bits/stdc++.h>

#define ll long long int

using namespace std;

void swap(int \*a, int \*b)

{

\*a^=\*b;

\*b^=\*a;

\*a^=\*b;

}

int Partition(int arr[], int l, int r)

{

int temp,pindex,pivot;

pindex=l;

//the following {} is for randomizing the pivot

{

srand(time(NULL));

temp = l + ( rand() % ( r - l + 1 ) );

swap(arr[temp],arr[r]);

}

pivot=r;

int i;

for(i=l;i<r;i++)

{

if(arr[i]<=arr[pivot])

{

swap(arr[i],arr[pindex]);

pindex++;

}

}

swap(arr[pindex],arr[r]);

return pindex;

}

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

{

if(l<r)

{

int pindex=Partition(arr,l,r);

QuickSort(arr,l,pindex-1);

QuickSort(arr,pindex+1,r);

}

}

int main()

{

int n,i;

cout << "Enter size of array : ";

cin >> n;

int arr[n];

cout << "Enter array elements : \n";

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

{

cin >> arr[i];

}

QuickSort(arr,0,n-1);

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

{

cout << arr[i] << ' ';

}

return 0;

}