

**DAA PRACTICAL FILE**

**2022-2023**

**(IV SEMESTER)**

**NAME: Gautam Deopa**

**SECTION: F**

**CLASS ROLL NUMBER: 05**

**UNIVERSITY ROLL NUMBER: 2016755**

**WEEK 1**

1.

#include <iostream>

using namespace std;

int main() {

int T;

cin >> T;

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

int n;

cin >> n;

int A[1000];

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

cin >> A[j];

}

int flag = 0, comp = 0, key;

cin >> key;

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

comp++;

if (A[k] == key) {

flag++;

cout << "Present ";

break;

}

}

if (flag == 0)

cout << "Not present ";

cout << comp << endl;

}

}



2.

#include <iostream>

using namespace std;

int main() {

int T;

cin >> T;

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

int n;

cin >> n;

int A[1000];

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

cin >> A[j];

int flag = 0, comp = 0, key, mid, lower = 0, greater = n - 1

cin >> key;

while (greater >= lower) {

comp++;

mid = lower + (greater - lower) / 2;

if (A[mid] == key) {

cout << "Present ";

flag++; break;

}

else if (key > A[mid])

lower = mid + 1;

else

greater = mid - 1;

}

if (flag == 0) cout << "Not Present ";

cout << comp << endl;

}

}



**WEEK 2**

1.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int count(int A[], int key, int n) {

int \*l = lower\_bound(A, A + n, key);

if (l == (A + n) || \*l != key)

return 0;

int \*f = upper\_bound(l, A + n, key);

return f - l;

}

int main() {

int T, n;

cin >> T;

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

cin >> n;

int lb = 0, ub = n - 1;

int A[1000];

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

cin >> A[i];

int key, c;

cin >> key;

c = count(A, key, n);

if (c == 0)

cout << "Key not present" << endl;

cout << key << " - " << c << endl;

}

}



2.

#include <iostream>

using namespace std;

int main() {

int T;

cin >> T;

for (int m = 0; m < T; m++) {

int A[1000];

int n;

cin >> n;

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

cin >> A[i];

int i, j, k;

int flag = 0;

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

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

int sum = A[i] + A[j];

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

if (sum == A[k]) {

flag = 1;

cout << i << " " << j << " " << k << endl;

break;

}

}

}

}

if (flag == 0)

cout << "Sequence not found" << endl;

}

}



3.

#include <iostream>

using namespace std;

int main() {

int T;

cin >> T;

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

int n;

cin >> n;

int A[1000];

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

cin >> A[j];

int key;

cin >> key;

int count = 0;

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

for (int k = j + 1; k < n; k++) {

if (A[j] - A[k] == key || A[k] - A[j] == key)

count++;

}

}

cout << count << endl;

cout << "-----------------------------" << endl;

}

}



**WEEK 3**

1.

#include <iostream>

using namespace std;

void Insert\_Sort(int[], int);

int main() {

int T;

cin >> T;

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

{

int n;

cin >> n;

int A[1000];

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

cin >> A[j];

Insert\_Sort(A, n);

}

}

void Insert\_Sort(int A[], int n) {

int temp, comp = 0, shift = 0;

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

temp = A[i];

int j = i - 1;

while (j >= 0 && A[j] > temp){

comp++;

A[j + 1] = A[j];

shift++; j--;

}

A[j + 1] = temp;

shift++;

}

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

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

cout << "\ncomparisons = " << comp

<< endl << "shifts = " << shift << endl;

}



2.

#include <iostream>

using namespace std;

void Sel\_Sort(int[], int);

int main()

{

int T;

cin >> T;

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

{

int n;

cin >> n;

int A[1000];

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

{

cin >> A[j];

}

Sel\_Sort(A, n);

}

}

void Sel\_Sort(int A[], int n)

{

int comp = 0, swaps = 0;

int min, temp = 0;

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

{

min = i;

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

{

comp++;

if (A[min] > A[j])

{

min = j;

}

}

swaps++;

swap(A[min], A[i]);

}

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

{

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

}

cout << "\ncomparisons = " << comp << endl

<< "swaps = " << swaps << endl;

}



3.

#include <iostream>

using namespace std;

void Quick\_Sort(int[], int, int);

int main()

{

int T;

cin >> T;

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

{

int n;

cin >> n;

int A[1000];

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

cin >> A[j];

Quick\_Sort(A, 0, n - 1);

int flag = 0;

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

{

if (A[j] == A[j + 1])

{

flag = 1;

cout << "YES" << endl;

break;

}

}

if (flag == 0)

cout << "NO" << endl;

}

}

void Quick\_Sort(int a[], int lb, int ub)

{

int i = lb, j = ub, key = a[lb];

if (i > j)

return;

while (i < j)

{

while (key >= a[i] && i < j)

i++;

while (key < a[j])

j--;

if (i < j)

{

int t = a[i];

a[i] = a[j];

a[j] = t;

}

}

a[lb] = a[j];

a[j] = key;

Quick\_Sort(a, lb, j - 1);

Quick\_Sort(a, j + 1, ub);

}



**WEEK 4**

1.

#include <iostream>

using namespace std;

int comp = 0;

int merge(int arr[], int l, int mid, int h) {

int count = 0;

int i = l, j = mid + 1;

int temp[1000];

int k = 0;

while (i <= mid && j <= h) {

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

temp[k++] = arr[i++];

else {

temp[k++] = arr[j++];

count += mid - i + 1;

}

comp++;

}

for (; i <= mid;)

temp[k++] = arr[i++];

for (; j <= h;)

temp[k++] = arr[j++];

for (int f = 0; f < k; f++)

arr[f + l] = temp[f];

return count;

}

int merge\_sort(int arr[], int l, int h)

{

int inversion = 0;

if (l < h)

{

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

inversion += merge\_sort(arr, l, mid);

inversion += merge\_sort(arr, mid + 1, h);

inversion += merge(arr, l, mid, h);

}

return inversion;

}

int main()

{

int T;

cin >> T;

while (T--)

{

int n;

cin >> n;

int arr[1000];

int prev\_comp = comp;

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

cin >> arr[i];

int inv = merge\_sort(arr, 0, n - 1);

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

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

cout << endl

<< "comparisons = " << comp - prev\_comp << endl

<< "inversions = " << inv << endl;

}

return 0;

}



3.

#include <iostream>

using namespace std;

void merge(int arr[], int l, int mid, int h)

{

int count = 0;

int i = l, j = mid + 1;

int temp[1000];

int k = 0;

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

{

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

temp[k++] = arr[i++];

else

{

temp[k++] = arr[j++];

count += mid - i + 1;

}

}

for (; i <= mid;)

temp[k++] = arr[i++];

for (; j <= h;)

temp[k++] = arr[j++];

for (int f = 0; f < k; f++)

arr[f + l] = temp[f];

}

void merge\_sort(int arr[], int l, int h)

{

if (l < h)

{

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

merge\_sort(arr, l, mid);

merge\_sort(arr, mid + 1, h);

merge(arr, l, mid, h);

}

}

int main()

{

int T;

cin >> T;

while (T--)

{

int n;

cin >> n;

int arr[1000];

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

cin >> arr[i];

int k;

cin >> k;

merge\_sort(arr, 0, n - 1);

int flag = 0;

cout << arr[k - 1] << endl;

}

}



**WEEK 5**

1.

#include <iostream>

using namespace std;

void countSort(char[], int);

int main()

{

int T;

cin >> T;

while (T--)

{

int n;

cin >> n;

char arr[1000];

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

cin >> arr[i];

countSort(arr, n);

}

}

void countSort(char arr[], int n)

{

int count[26] = {0};

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

count[arr[i] - 97]++;

int max = 0;

char res = '$';

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

{

if (count[i] > max)

{

max = count[i];

res = i + 97;

}

}

if (max == 1)

cout << "No duplicate found" << endl;

else

cout << res << " - " << max << endl;

}



2.

#include <iostream>

using namespace std;

void Quick\_Sort(int[], int, int);

int main() {

int T;

cin >> T;

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

int n;

cin >> n;

int A[1000];

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

cin >> A[j];

Quick\_Sort(A, 0, n - 1);

int x = 0, y = n - 1, sum;

cin >> sum;

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

if (A[x] + A[y] == sum) {

cout << A[x] << " " << A[y] << endl;

break;

}

else if (A[x] + A[y] > sum)

y--;

else if (A[x] + A[y] < sum)

x++;

if (x > y)

{

cout << "No such Element Exist";

break;

}

}

}

}

void Quick\_Sort(int a[], int lb, int ub)

{

int i = lb, j = ub, key = a[lb];

if (i > j)

return;

while (i < j)

{

while (key >= a[i] && i < j)

i++;

while (key < a[j])

j--;

if (i < j)

{

int t = a[i];

a[i] = a[j];

a[j] = t;

}

}

a[lb] = a[j];

a[j] = key;

Quick\_Sort(a, lb, j - 1);

Quick\_Sort(a, j + 1, ub);

}



3.

#include <iostream>

using namespace std;

int main() {

int m;

cin >> m;

int A[100];

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

cin >> A[i];

int n;

cin >> n;

int B[100];

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

cin >> B[i];

int p = 0, q = 0;

while (p < m && q < n)

{

if (A[p] == B[q])

{

cout << A[p] << " ";

p++;

q++;

}

else if (A[p] < B[q])

p++;

else

q++;

}

}

