1. Given an array of nonnegative integers, design a linear algorithm and implement it using a programto find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(n), where n is the size of input)

#include <stdio.h>

#include <stdbool.h>

int main(){

int size, count=0, i, target;

bool flag=false;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

printf("Enter elements in array ");

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

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

printf("Enter element to search in array ");

scanf("%d", &target);

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

count++;

if(arr[i]==target){

flag=true;

break;

}

}

if(flag==true)

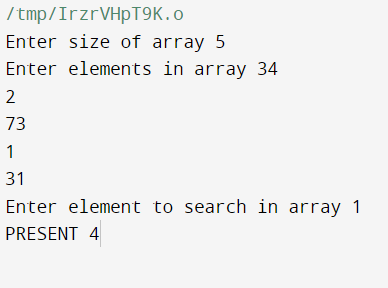
printf("PRESENT %d", count);

else

printf("NOT PRESENT");

return 0;

}



II. Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(logn), where n is the size of input).

#include <stdio.h>

void search(int arr[], int target, int \*count, int l, int r){

if(l>r){

printf("NOT PRESENT ");

return;

}

int mid = (l+r)/2;

(\*count)++;

if(arr[mid]==target){

printf("PRESENT ");

return;

}

else if(arr[mid]<target){

search(arr, target, count, mid+1, r);

}

else if(arr[mid]>target){

search(arr, target, count, l, mid-1);

}

}

int main(){

int size, count=0, i, target, l, r;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

l=0, r=size-1;

printf("Enter elements in array ");

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

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

}

printf("Enter element to search in array ");

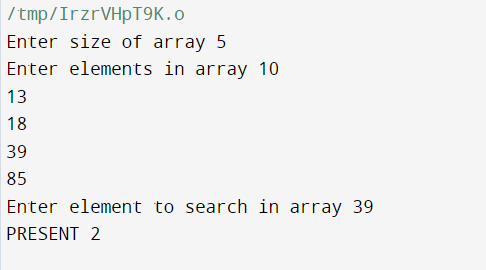
scanf("%d", &target);

search(arr, target, &count, l, r);

printf("%d", count);

return 0;

}



3.. Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n))

#include <stdio.h>

void search(int arr[], int target, int \*count, int l, int r, int \*start, int \*end, int size){

if(l>r){

return;

}

int mid = (l+r)/2;

(\*count)++;

if(arr[mid]==target){

if(arr[mid]==arr[mid-1] && mid!=0){

search(arr, target, count, l, mid-1, start, end, size);

}

else{

(\*start) = mid;

}

if(arr[mid]==arr[mid+1] && mid<size){

search(arr, target, count, l, mid-1, start, end, size);

}

else{

(\*end) = mid;

}

}

else if(arr[mid]<target){

search(arr, target, count, mid+1, r, start, end, size);

}

else if(arr[mid]>target){

search(arr, target, count, l, mid-1, start , end, size);

}

}

int main(){

int size, count=0, i, target, l, r, start=-1, end=-1, t\_case, t;

printf("Enter number of test cases ");

scanf("%d", &t\_case);

for(t=0; t<t\_case; t++){

start=-1, end=-1;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

l=0, r=size-1;

printf("Enter elements in array ");

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

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

}

printf("Enter element to search in array ");

scanf("%d", &target);

search(arr, target, &count, l, r, &start, &end, size);

if(start==-1)

printf("NOT PRESENT\n");

else{

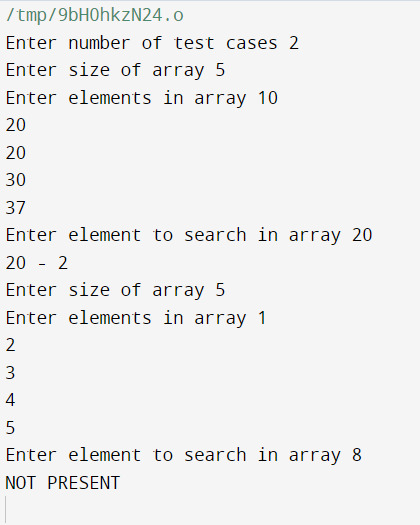
printf("%d - %d\n", target, (end-start+1));

}

}

return 0;

}



4 …Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k].

#include <stdio.h>

#include <stdbool.h>

bool search(int arr[], int target, int l, int r, int \*ind){

if(l>r){

return false;

}

int mid = (l+r)/2;

if(arr[mid]==target){

\*ind = mid;

return true;

}

else if(arr[mid]<target){

search(arr, target, mid+1, r, ind);

}

else if(arr[mid]>target){

search(arr, target, l, mid-1, ind);

}

}

int main(){

int t\_case, l, r, i, j, size, index=-1, k;

printf("Enter number of test cases ");

scanf("%d", &t\_case);

for(i=0; i<t\_case; i++){

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

l=0;

r = size-1;

index=-1;

printf("Enter elements in array ");

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

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

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

for(k=j+1; k<size; k++){

if(search(arr, arr[j]+arr[k], l, r, &index)==true){

printf("%d %d %d\n", j+1, k+1, index+1);

break;

}

}

}

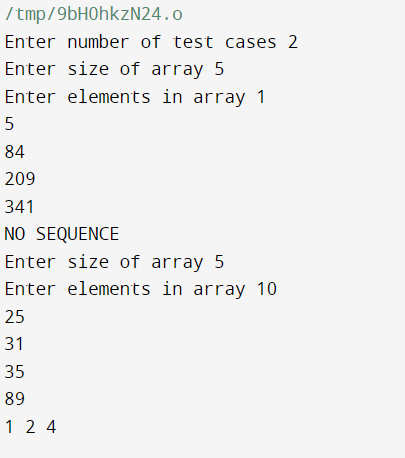
if(index==-1)

printf("NO SEQUENCE\n");

}

return 0;

}



5.. Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts ( shifts - total number of times the array elements are shifted from their place) required forsorting the array.

#include <stdio.h>

int main(){

int t\_case, size, i, j, t, comparison=0, shifts=0, temp, iter;

printf("enter number of test cases ");

scanf("%d", &t\_case);

for(t=0; t<t\_case; t++){

comparison=0;

shifts=0;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

printf("ENter elements ");

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

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

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

temp = arr[i];

iter = i-1;

while(temp<arr[iter] && iter>=0){

comparison++;

arr[iter+1] = arr[iter];

iter = iter-1;

}

arr[iter+1] = temp;

shifts++;

}

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

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

}

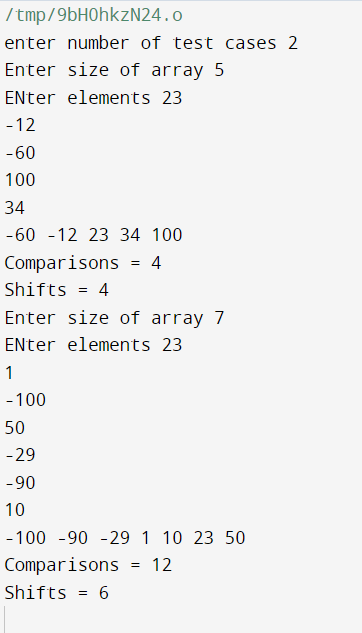
printf("\n");

printf("Comparisons = %d\n", comparison);

printf("Shifts = %d\n", shifts);

}

}



6..Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required

#include <stdio.h>

int main(){

int t\_case, i, j, size, t, min, swap, comparison, temp;

printf("Enter no. of test cases ");

scanf("%d", &t\_case);

for(t=0; t<t\_case; t++){

comparison=0;

swap=0;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

printf("Enter array elements ");

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

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

}

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

min=i;

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

comparison++;

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

min = j;

}

if(min!=i){

temp = arr[min];

arr[min] = arr[i];

arr[i] = temp;

swap++;

}

}

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

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

}

printf("\n");

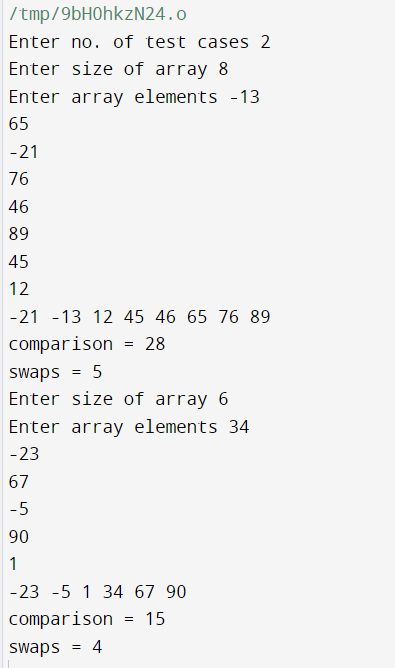
printf("comparison = %d\n", comparison);

printf("swaps = %d\n", swap);

}

return 0;

}



7…Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n))

#include <stdio.h>

#include <stdbool.h>

void merge(int \*arr, int l, int r){

int i;

int mid = (l+r)/2;

int len1 = mid-l+1;

int len2 = r-mid;

int arr1[len1];

int arr2[len2];

int index = l;

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

arr1[i] = arr[index++];

}

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

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

}

int index1 = 0;

int index2 = 0;

index = l;

while(index1<len1 && index2<len2){

if(arr1[index1]<arr2[index2]){

arr[index++] = arr1[index1++];

}

else{

arr[index++] = arr2[index2++];

}

}

while(index1<len1)

arr[index++] = arr1[index1++];

while(index2<len2)

arr[index++] = arr2[index2++];

}

void mergeSort(int \*arr, int l, int r){

int mid = (l+r)/2;

if(l>=r)

return;

mergeSort(arr, l, mid);

mergeSort(arr, mid+1, r);

merge(arr, l, r);

}

int main(){

int t\_case, i, j, size, t, min, swap, comparison, temp, l, r;

bool flag;

printf("Enter no. of test cases ");

scanf("%d", &t\_case);

for(t=0; t<t\_case; t++){

flag=false;

comparison=0;

swap=0;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

l=0;

r=size-1;

printf("Enter array elements ");

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

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

}

mergeSort(&arr, l, r);

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

if(arr[i]==arr[i+1]){

flag=true;

break;

}

}

if(flag==true)

printf("YES\n");

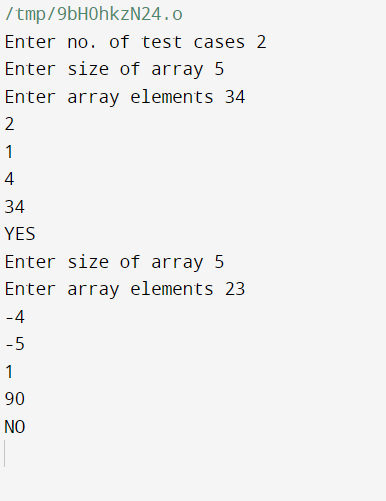
else

printf("NO\n");

}

return 0;

}



8…Given an unsorted array of integers, design an algorithm and implement it using a program to sortan array of elements by dividing the array into two subarrays and combining these subarrays aftersorting each one of them. Your program should also find number of comparisons and inversions during sorting the array

#include <stdio.h>

void merge(int \*arr, int l, int r, int\*comparison){

int i;

int mid = (l+r)/2;

int len1 = mid-l+1;

int len2 = r-mid;

int arr1[len1];

int arr2[len2];

int index = l;

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

arr1[i] = arr[index++];

}

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

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

}

int index1 = 0;

int index2 = 0;

index = l;

while(index1<len1 && index2<len2){

(\*comparison)++;

if(arr1[index1]<arr2[index2]){

arr[index++] = arr1[index1++];

}

else{

arr[index++] = arr2[index2++];

}

}

while(index1<len1)

arr[index++] = arr1[index1++];

while(index2<len2)

arr[index++] = arr2[index2++];

}

void mergeSort(int \*arr, int l, int r, int \*comparison, int \*inversion){

int mid = (l+r)/2;

if(l>=r)

return;

(\*inversion)++;

mergeSort(arr, l, mid, comparison, inversion);

mergeSort(arr, mid+1, r, comparison, inversion);

merge(arr, l, r, comparison);

}

int main(){

int t\_case, i, j, size, t, min, inversion, comparison, temp, l, r;

printf("Enter no. of test cases ");

scanf("%d", &t\_case);

for(t=0; t<t\_case; t++){

inversion=0;

comparison=0;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

l=0;

r=size-1;

printf("Enter array elements ");

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

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

}

mergeSort(&arr, l, r, &comparison, &inversion);

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

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

printf("\n");

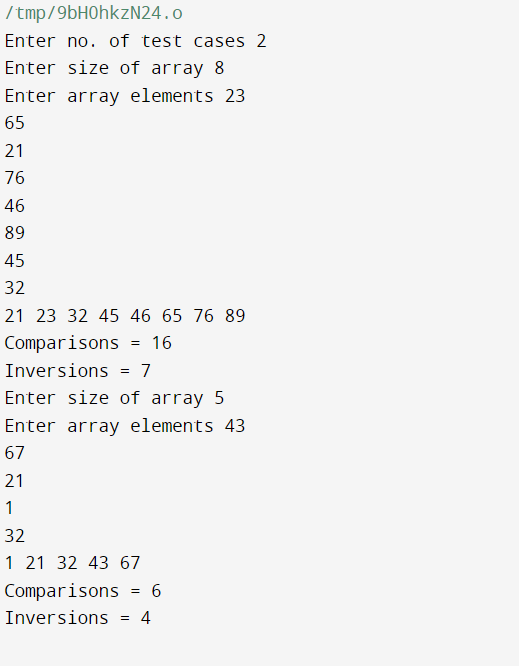
printf("Comparisons = %d\n", comparison);

printf("Inversions = %d\n", inversion);

}

return 0;

}



9…Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by partitioning the array into two subarrays based on a pivot element such that one of the sub array holds values smaller than the pivot element while another subarray holds values greater than the pivot element. Pivot element should be selected randomly from the array. Your program should also find number of comparisons and swaps required for sorting the array.

#include <stdio.h>

int partition(int arr[], int l, int r, int \*comparison, int \*swap) {

int pivot = arr[l];

int cnt = 0, i, pivotIndex, temp, j;

for (i = l + 1; i <= r; i++) {

(\*comparison)++;

if (pivot > arr[i])

cnt++;

}

pivotIndex = l + cnt;

(\*swap)++;

temp = arr[l];

arr[l] = arr[pivotIndex];

arr[pivotIndex] = temp;

i = l;

j = r;

while (i < pivotIndex && j > pivotIndex) {

while (arr[i] < pivot)

i++;

while (arr[j] > pivot)

j--;

if (i < pivotIndex && j > pivotIndex) {

(\*swap)++;

(\*swap)++;

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

return pivotIndex;

}

void quickSort(int arr[], int l, int r, int \*comparison, int \*swap) {

if (l >= r)

return;

int p = partition(arr, l, r, comparison, swap);

quickSort(arr, l, p - 1, comparison, swap);

quickSort(arr, p + 1, r, comparison, swap);

}

int main() {

int t\_case, i, size, t, swap, comparison, l, r;

printf("Enter no. of test cases ");

scanf("%d", &t\_case);

for (t = 0; t < t\_case; t++) {

swap = 0;

comparison = 0;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

l = 0;

r = size - 1;

printf("Enter array elements ");

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

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

}

quickSort(arr, l, r, &comparison, &swap);

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

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

printf("\n");

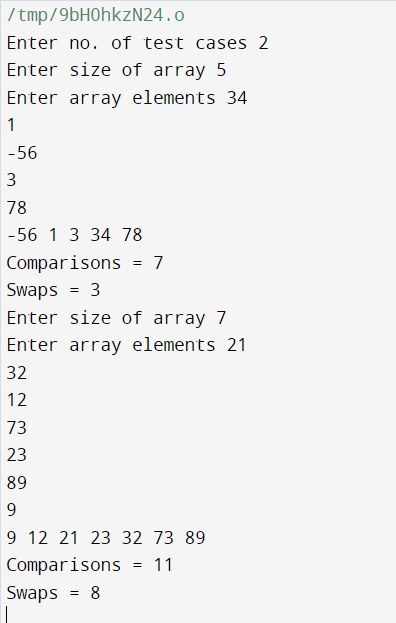
printf("Comparisons = %d\n", comparison);

printf("Swaps = %d\n", swap);

}

return 0;

}



10..Given an unsorted array of integers, design an algorithm and implement it using a program to find Kth smallest or largest element in the array. (Worst case Time Complexity = O(n))

#include <stdio.h>

#include <limits.h>

int main() {

int t\_case, i, size, t, greatest = INT\_MIN, pos, k, count;

printf("Enter no. of test cases ");

scanf("%d", &t\_case);

for (t = 0; t < t\_case; t++) {

count = 0;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

int result[size];

printf("Enter array elements ");

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

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

if (arr[i] > greatest)

greatest = arr[i];

}

printf("Enter k ");

scanf("%d", &k);

if (k > size) {

printf("NOT PRESENT\n");

continue;

}

int temp[greatest + 1];

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

temp[i] = 0;

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

temp[arr[i]]++;

}

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

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

}

for (i = size - 1; i >= 0; i--) {

result[temp[arr[i]] - 1] = arr[i];

temp[arr[i]]--;

}

printf("\n");

for (i = size-1; i > 0; i--) {

if (result[i] > result[i - 1]) {

count++;

}

if (count == k) {

printf("Kth greatest element is %d\n", result[i]);

break;

}

}

if(count+1 ==k && result[0]<result[1]){

printf("Kth greatest element is %d\n", result[0]);

}

else {

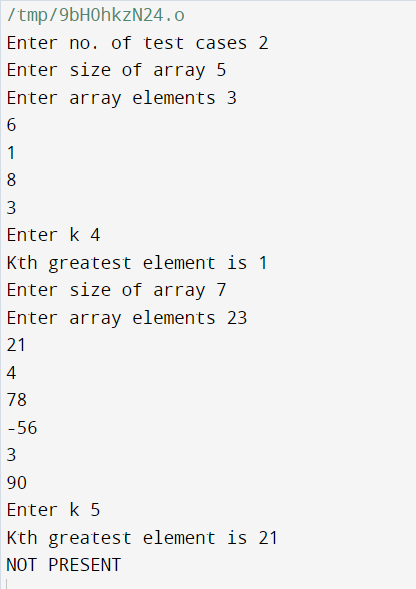
printf("NOT PRESENT\n");

}

}

return 0;

}



11. Given an unsorted array of alphabets containing duplicate elements. Design an algorithm and implement it using a program to find which alphabet has maximum number of occurrences and print it.

#include <stdio.h>

int main() {

int t\_case, i, size, t, greatest;

printf("Enter no. of test cases ");

scanf("%d", &t\_case);

for (t = 0; t < t\_case; t++) {

printf("Enter size of array ");

scanf("%d", &size);

getchar();

char arr[size];

int count[26] = {0};

printf("Enter array elements ");

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

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

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

count[arr[i] - 'a']++;

greatest = 0;

for (i = 1; i < 26; i++) {

if (count[i] > count[greatest])

greatest = i;

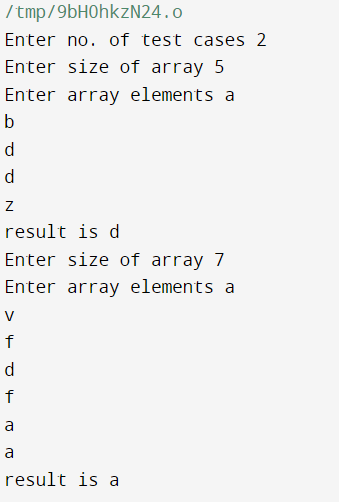
}

printf("result is %c\n", 'a' + greatest);

}

return 0;

}



12.. Given an unsorted array of integers, design an algorithm and implement it using a program to findwhether two elements exist such that their sum is equal to the given key element. (Time Complexity = O(n log n))

#include <stdio.h>

#include <stdbool.h>

bool search(int arr[], int second, int l, int r, bool\* flag) {

if (l > r) {

return false;

}

int mid = (l + r) / 2;

if (arr[mid] == second) {

\*flag = true;

return true;

} else if (arr[mid] < second) {

return search(arr, second, mid + 1, r, flag);

} else if (arr[mid] > second) {

return search(arr, second, l, mid - 1, flag);

}

}

int main() {

int t\_case, i, size, t, first, second, target;

bool flag;

printf("Enter no. of test cases ");

scanf("%d", &t\_case);

for (t = 0; t < t\_case; t++) {

flag = false;

printf("Enter size of array ");

scanf("%d", &size);

int arr[size];

printf("Enter array elements ");

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

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

printf("Enter target number ");

scanf("%d", &target);

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

second = target - arr[i];

search(arr, second, i + 1, size - 1, &flag);

if (flag) {

first = arr[i];

break;

}

}

if (flag)

printf("%d %d\n", first, second);

else

printf("NOT PRESENT\n");

}

return 0;

}

