1.Given an array Arr of size N, print second largest distinct element from an array. **Find the second largest without sorting.**

#include <stdio.h>

int secondLargestDistinct(int arr[], int n) {

int largest = -1, secondLargest = -1;

// Iterate through the array

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

if (arr[i] > largest) {

secondLargest = largest; // Update secondLargest to previous largest

largest = arr[i]; // Update largest to current element

} else if (arr[i] > secondLargest && arr[i] != largest) {

secondLargest = arr[i]; // Update secondLargest if distinct and larger

}

}

return secondLargest;

}

int main() {

int n;

printf("Enter the size of the array: ");

scanf("%d", &n);

int arr[n];

printf("Enter the elements of the array: ");

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

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

}

int secondLargest = secondLargestDistinct(arr, n);

if (secondLargest == -1) {

printf("There are less than two distinct elements in the array.\n");

} else {

printf("The second largest distinct element is %d\n", secondLargest);

}

return 0;

}

2.Given an array Arr of N positive integers and another number X. **Determine whether or not there exist two elements in Arr whose sum is exactly X.**[Without Sorting]

#include <stdio.h>

#define MAX\_SIZE 100 // Adjust maximum array size if needed

int find\_pair\_sum(int arr[], int n, int x) {

int seen[MAX\_SIZE] = {0}; // Initialize a hash table to track seen elements

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

int complement = x - arr[i];

if (seen[complement] == 1) { // Check if the complement exists in the hash table

return 1; // Pair found

}

seen[arr[i]] = 1; // Mark the current element as seen

}

return 0; // Pair not found

}

int main() {

int n, x;

printf("Enter the size of the array: ");

scanf("%d", &n);

int arr[n];

printf("Enter the elements of the array: ");

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

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

}

printf("Enter the target sum: ");

scanf("%d", &x);

if (find\_pair\_sum(arr, n, x)) {

printf("Yes, there exists a pair with sum %d\n", x);

} else {

printf("No, there does not exist a pair with sum %d\n", x);

}

return 0;

}

**3.First and last occurrences of x**

Given a sorted array arr containing n elements with possibly some duplicate, the task is to find the first and last occurrences of an element x in the given array.

#include <stdio.h>

int findFirstOccurrence(int arr[], int n, int x) {

int low = 0, high = n - 1;

while (low <= high) {

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

if (arr[mid] == x) {

if (mid == 0 || arr[mid - 1] != x) { // Check for first occurrence

return mid;

} else {

high = mid - 1; // Search for earlier occurrence

}

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

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1; // Not found

}

int findLastOccurrence(int arr[], int n, int x) {

int low = 0, high = n - 1;

while (low <= high) {

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

if (arr[mid] == x) {

if (mid == n - 1 || arr[mid + 1] != x) { // Check for last occurrence

return mid;

} else {

low = mid + 1; // Search for later occurrence

}

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

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1; // Not found

}

int main() {

int n, x;

printf("Enter the size of the array: ");

scanf("%d", &n);

int arr[n];

printf("Enter the elements of the array: ");

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

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

}

printf("Enter the element to search: ");

scanf("%d", &x);

int first = findFirstOccurrence(arr, n, x);

int last = findLastOccurrence(arr, n, x);

if (first == -1) {

printf("Element not found in the array.\n");

} else {

printf("First occurrence: %d\nLast occurrence: %d\n", first, last);

}

return 0;

}