**1.Swapping of two Numbers** by

a)Call By Value

b)Call By Reference

a)

#include <stdio.h>

void swap(int x, int y) {

int temp = x;

x = y;

y = temp;

printf("Inside swap function: x = %d, y = %d\n", x, y);

}

int main() {

int a = 10, b = 20;

printf("Before swapping in main: a = %d, b = %d\n", a, b);

swap(a, b); // Call by value

printf("After swapping in main: a = %d, b = %d\n", a, b);

return 0;

}

b)

#include <stdio.h>

void swap(int \*x, int \*y) {

int temp = \*x;

\*x = \*y;

\*y = temp;

printf("Inside swap function: x = %d, y = %d\n", \*x, \*y);

}

int main() {

int a = 10, b = 20;

printf("Before swapping in main: a = %d, b = %d\n", a, b);

swap(&a, &b); // Call by reference

printf("After swapping in main: a = %d, b = %d\n", a, b);

return 0;

}

**2.Find duplicates in an array**

Given an array a of size N which contains elements from 0 to N-1, you need to find all the elements occurring more than once in the given array. Return the answer in ascending order. If no such element is found, return list containing [-1].

Example 1:

Input:

N = 4

a[] = {0,3,1,2}

Output:

-1

Explanation: There is no repeating element in the array. Therefore output is -1.

Example 2:

Input:

N = 5

a[] = {2,3,1,2,3}

Output:

2 3

Explanation: 2 and 3 occur more than once in the given array

#include <stdio.h>

int \*find\_duplicates(int arr[], int n) {

// Create a frequency array to store the count of each element

int freq[n] = {0}; // Initialize all frequencies to 0

// Iterate through the array and increment the frequency of each element

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

freq[arr[i]]++;

}

// Create a result array to store the duplicates

int \*result = (int \*)malloc(sizeof(int) \* n); // Allocate memory for result array

int result\_idx = 0; // Index for the result array

// Iterate through the frequency array and store elements with frequency greater than 1

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

if (freq[i] > 1) {

result[result\_idx++] = i; // Add the duplicate element to the result array

}

}

// If no duplicates found, return list containing [-1]

if (result\_idx == 0) {

result[0] = -1; // Set the first element to -1

result\_idx = 1; // Adjust the result index

}

// Return the result array with the duplicates

return result;

}

int main() {

int arr[] = {2, 3, 1, 2, 3};

int n = sizeof(arr) / sizeof(arr[0]);

int \*duplicates = find\_duplicates(arr, n);

// Print the duplicates in ascending order

printf("Duplicates: ");

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

if (duplicates[i] != -1) {

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

} else {

break; // Stop printing if -1 is encountered

}

}

printf("\n");

free(duplicates); // Free the allocated memory for the result array

return 0;

}

**3.Union of Two Sorted Arrays**

Union of two arrays can be defined as the common and distinct elements in the two arrays. Given two sorted arrays of size n and m respectively, find their union.

Example 1:

Input:

n = 5, arr1[] = {1, 2, 3, 4, 5}

m = 3, arr2 [] = {1, 2, 3}

Output: 1 2 3 4 5

Explanation: Distinct elements including

both the arrays are: 1 2 3 4 5.

Example 2:

Input:

n = 5, arr1[] = {2, 2, 3, 4, 5}

m = 5, arr2[] = {1, 1, 2, 3, 4}

Output: 1 2 3 4 5

Explanation: Distinct elements including

both the arrays are: 1 2 3 4 5

#include <stdio.h>

void printUnion(int arr1[], int arr2[], int n, int m) {

int i = 0, j = 0;

while (i < n && j < m) {

// If current elements are equal, print one of them and move both pointers

if (arr1[i] == arr2[j]) {

printf("%d ", arr1[i]);

i++;

j++;

}

// If arr1[i] is smaller, move pointer i

else if (arr1[i] < arr2[j]) {

printf("%d ", arr1[i]);

i++;

}

// If arr2[j] is smaller, move pointer j

else {

printf("%d ", arr2[j]);

j++;

}

}

// Print remaining elements of arr1, if any

while (i < n) {

printf("%d ", arr1[i]);

i++;

}

// Print remaining elements of arr2, if any

while (j < m) {

printf("%d ", arr2[j]);

j++;

}

}

int main() {

int arr1[] = {1, 2, 3, 4, 5};

int arr2[] = {1, 2, 3};

int n = sizeof(arr1) / sizeof(arr1[0]);

int m = sizeof(arr2) / sizeof(arr2[0]);

printf("Union of two arrays is: ");

printUnion(arr1, arr2, n, m);

printf("\n");

return 0;

}