**QUIZ 7**

**USE POINTERS**

**1.Swapping of two Numbers** by   
a)Call By Value  
b)Call By Reference

#include <stdio.h>

void swapByValue(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int num1 = 5, num2 = 10;

printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

swapByValue(&num1, &num2);

printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

return 0;

}

OUTPUT:

Before swapping: num1 = 5, num2 = 10

After swapping: num1 = 10, num2 = 5

#include <stdio.h>

void swapByReference(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int num1 = 5, num2 = 10;

printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

swapByReference(&num1, &num2);

printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

return 0;

}

OUTPUT:

Before swapping: num1 = 5, num2 = 10

After swapping: num1 = 10, num2 = 5

**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>

void findDuplicates(int arr[], int n) {

int \*count = (int \*)malloc(n \* sizeof(int));

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

count[i] = 0;

}

int hasDuplicates = 0;

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

count[arr[i]]++;

if (count[arr[i]] == 2) {

hasDuplicates = 1;

}

}

if (hasDuplicates) {

printf("Duplicates: ");

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

if (count[i] == 2) {

printf("%d ", i);

}

}

printf("\n");

} else {

printf("-1\n");

}

free(count);

}

int main() {

int N1 = 4;

int a1[] = {0, 3, 1, 2};

printf("Example 1:\n");

findDuplicates(a1, N1);

int N2 = 5;

int a2[] = {2, 3, 1, 2, 3};

printf("\nExample 2:\n");

findDuplicates(a2, N2);

return 0;

}

OUTPUT:

Example 1:

-1

Example 2:

Duplicates: 2 3

**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 n, int arr2[], int m) {

int i = 0, j = 0;

printf("Union of the arrays: ");

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

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

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

i++;

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

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

j++;

} else

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

i++;

j++;

}

}

while (i < n) {

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

i++;

}

while (j < m) {

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

j++;

}

printf("\n");

}

int main() {

int n1 = 5, arr1[] = {1, 2, 3, 4, 5};

int m1 = 3, arr2[] = {1, 2, 3};

printf("Example 1:\n");

printUnion(arr1, n1, arr2, m1);

int n2 = 5, arr3[] = {2, 2, 3, 4, 5};

int m2 = 5, arr4[] = {1, 1, 2, 3, 4};

printf("\nExample 2:\n");

printUnion(arr3, n2, arr4, m2);

return 0;

}

OUTPUT:

Example 1:

Union of the arrays: 1 2 3 4 5

Example 2:

Union of the arrays: 1 2 3 4 5