ISA 3

Name: Gauresh Rekdo

Roll no: 17

Implement function swapping of array using C. Consider pass by reference as argument to function. If threads are invoked then will this implementation require semaphore to effectively handle the scenario. Justify your answer with solution implemented using C code. Also discuss whether this concept is used in any real applications

# C Code for Array Swapping:

#include <stdio.h>

// Function to swap two arrays by reference

void swap\_arrays(int \*\*arr1, int \*\*arr2) {

int \*temp = \*arr1;

\*arr1 = \*arr2;

\*arr2 = temp;

}

int main() {

// Define two arrays

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

int arr2[] = {4, 5, 6};

// Define pointers to the arrays

int \*p1 = arr1;

int \*p2 = arr2;

// Print arrays before swapping

printf("Before swapping:\nArray 1: ");

for (int i = 0; i < 3; i++) {

printf("%d ", p1[i]);

}

printf("\nArray 2: ");

for (int i = 0; i < 3; i++) {

printf("%d ", p2[i]);

}

// Call the function to swap the arrays

swap\_arrays(&p1, &p2);

// Print arrays after swapping

printf("\n\nAfter swapping:\nArray 1: ");

for (int i = 0; i < 3; i++) {

printf("%d ", p1[i]);

}

printf("\nArray 2: ");

for (int i = 0; i < 3; i++) {

printf("%d ", p2[i]);

}

return 0;

}

#Semaphore Justification for Threading:

When threads swap arrays, race conditions can occur if multiple threads modify the arrays simultaneously, causing inconsistent results. A semaphore ensures that only one thread performs the swap at a time, maintaining data integrity.

#Real World Application:

In multithreaded memory management, semaphores prevent simultaneous access to shared buffers during data swaps, ensuring memory integrity in systems handling large-scale data exchanges or concurrent processing.