6. Write a C program to implement the deadlock detection algorithm for a system with 3 processes and 3 resource instances and the resource matrices are given below.

Max Matrix Allocation Matrix

3 6 8 3 3 3

4 3 3 2 0 3

3 4 4 1 2 4

#include <stdio.h>

#include <stdbool.h>

#define NUM\_PROCESSES 3

#define NUM\_RESOURCES 3

// Function to check if a process can finish

bool can\_finish(int process, int available[], int max[][NUM\_RESOURCES], int allocation[][NUM\_RESOURCES])

{

// Check if there are enough resources available for the process to finish

for (int i = 0; i < NUM\_RESOURCES; i++)

{

if (available[i] < max[process][i] - allocation[process][i])

{

return false;

}

}

return true;

}

// Function to detect deadlocks in the system

bool detect\_deadlock(int allocation[][NUM\_RESOURCES], int max[][NUM\_RESOURCES], int available[])

{

bool marked[NUM\_PROCESSES] = { false }; // Marked processes

int work[NUM\_RESOURCES]; // Work array

for (int i = 0; i < NUM\_RESOURCES; i++)

{

work[i] = available[i];

}

int finish\_count = 0; // Number of finished processes

while (finish\_count < NUM\_PROCESSES)

{

bool found = false; // Flag to indicate if a process was found

for (int i = 0; i < NUM\_PROCESSES; i++)

{

if (!marked[i] && can\_finish(i, work, max, allocation))

{

// Process i can finish, so mark it and release its resources

marked[i] = true;

found = true;

finish\_count++;

for (int j = 0; j < NUM\_RESOURCES; j++)

{

work[j] += allocation[i][j];

}

}

}

// If no process can finish, there is a deadlock

if (!found)

{

return true;

}

}

// No deadlock found

return false;

}

int main()

{

int max[NUM\_PROCESSES][NUM\_RESOURCES] = {

{ 3, 6, 8 },

{ 4, 3, 3 },

{ 3, 4, 4 }

};

int allocation[NUM\_PROCESSES][NUM\_RESOURCES] = {

{ 3, 3, 3 },

{ 2, 0, 3 },

{ 1, 2, 4 }

};

int available[NUM\_RESOURCES] = { 0, 0, 0 }; // All resources initially available

if (detect\_deadlock(allocation, max, available))

{

printf("Deadlock detected\n");

}

else

{

printf("No deadlock detected\n");

}

return 0;

}

