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

1. 6 8 3 3 3
2. 3 3 2 0 3

3 4 4 1 2 4

The number of available resources is [1,2,0]. Determine if the system is in a deadlock state and identify the deadlocked processes.

Program:

#include <stdio.h>

int main() {

// Define the Max and Allocation matrices int max[3][3] = {{3, 6, 8}, {4, 3, 3}, {3, 4, 4}}; int allocation[3][3] = {{3, 3, 3}, {2, 0, 3}, {1, 2, 4}};

// Define the Available vector int available[3] = {1, 2, 0};

// Define the Work and Finish vectors int work[3], finish[3] = {0, 0, 0};

// Initialize the Work vector to the Available vector

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

work[i] = available[i];

}

// Initialize the Need matrix to the Max matrix minus the Allocation matrix int need[3][3]; for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) {

need[i][j] = max[i][j] - allocation[i][j];

}

}

// Detect deadlock by checking for a safe sequence int safe = 0; while (safe == 0) { safe = 1; for (int i = 0; i < 3; i++) { if (finish[i] == 0) {

int j;

for (j = 0; j < 3; j++) { if (need[i][j] > work[j]) {

break;

} }

if (j == 3) {

// Process i can complete

safe = 0; finish[i] = 1;

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

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

}

}

}

}

}

// Print the results int deadlock = 1;

printf("Deadlocked processes: "); for (int i = 0; i < 3; i++) { if (finish[i] == 0) { printf("%d ", i + 1); deadlock = 0;

}

}

if (deadlock == 1) {

printf("None");

}

printf("\n");

return 0; }

Output: s

