**Assignment 5**

#include <stdio.h>

#define MAX 4

// Adjacency matrix of Wait-For Graph

// wait\_for[i][j] = 1 means Pi is waiting for Pj

int wait\_for[MAX][MAX];

// Visited array for DFS

int visited[MAX], rec\_stack[MAX];

// Detect cycle using DFS

int is\_cyclic(int i) {

if (!visited[i]) {

visited[i] = 1;

rec\_stack[i] = 1;

for (int j = 0; j < MAX; j++) {

if (wait\_for[i][j]) {

if (!visited[j] && is\_cyclic(j))

return 1;

else if (rec\_stack[j])

return 1;

}

}

}

rec\_stack[i] = 0;

return 0;

}

int main() {

// Initialize Wait-For Graph based on the image

// P0 waits for R2 (held by P3) → P0 → P3

// P2 waits for R2 (held by P3) → P2 → P3

// P2 waits for R3 (held by P0) → P2 → P0

// P3 waits for R1 (held by P1 and P0) → P3 → P1 and P0

// Setting edges (Pi waits for Pj)

wait\_for[0][3] = 1; // P0 → P3

wait\_for[2][3] = 1; // P2 → P3

wait\_for[2][0] = 1; // P2 → P0

wait\_for[3][1] = 1; // P3 → P1

wait\_for[3][0] = 1; // P3 → P0

// Check for deadlock

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

visited[i] = 0;

rec\_stack[i] = 0;

}

int deadlock = 0;

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

if (is\_cyclic(i)) {

deadlock = 1;

break;

}

}

if (deadlock)

printf("🔴 System is in DEADLOCK state.\n");

else

printf("🟢 System is in a SAFE state.\n");

return 0;

}