**EXP NO:** 9 **DEADLOCK AVOIDANCE**

**DATE:19/2/25**

**PROGRAM:**

#include <stdio.h>

#include <stdbool.h>

#define P 5  // Number of processes

#define R 3  // Number of resources

// Function to find the safe sequence using Banker's Algorithm

bool isSafe(int processes[], int avail[], int max[][R], int allot[][R], int safeSeq[]) {

    int need[P][R];

    bool finish[P] = {false};

    int work[R];

    int count = 0;

    // Calculate the 'need' matrix

    for (int i = 0; i < P; i++) {

        for (int j = 0; j < R; j++) {

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

        }

    }

    // Initialize the work vector with available resources

    for (int i = 0; i < R; i++) {

        work[i] = avail[i];

    }

    // Find the safe sequence

    while (count < P) {

        bool found = false;

        for (int p = 0; p < P; p++) {

            if (!finish[p]) {

                int i;

                // Check if all resources needed by process p are available

                for (i = 0; i < R; i++) {

                    if (need[p][i] > work[i]) {

                        break;

                    }

                }

                // If all needs are met, allocate resources and mark process as finished

                if (i == R) {

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

                        work[j] += allot[p][j];

                    }

                    safeSeq[count++] = processes[p];

                    finish[p] = true;

                    found = true;

                }

            }

        }

        // If no process is found that can proceed, return false

        if (!found) {

            return false;

        }

    }

    return true;

}

int main() {

    int processes[] = {0, 1, 2, 3, 4}; // Process IDs

    // Available instances of resources

    int avail[] = {3, 3, 2}; // Example: 3 instances of resource 1, 3 of resource 2, 2 of resource 3

    // Maximum demand of each process for each resource

    int max[][R] = {

        {7, 5, 3},

        {3, 2, 2},

        {9, 0, 2},

        {2, 2, 2},

        {4, 3, 3}

    };

    // Allocation matrix (resources allocated to each process)

    int allot[][R] = {

        {0, 1, 0},

        {2, 0, 0},

        {3, 0, 2},

        {2, 1, 1},

        {0, 0, 2}

    };

    int safeSeq[P];

    // Check for the safe sequence

    if (isSafe(processes, avail, max, allot, safeSeq)) {

        printf("The SAFE Sequence is:\n");

        for (int i = 0; i < P; i++) {

            printf("P%d -> ", safeSeq[i]);

        }

        printf("\n");

    } else {

        printf("There is no safe sequence\n");

    }

    return 0;

}

**OUTPUT:**

The SAFE Sequence is:

P1 -> P3 -> P4 -> P0 -> P2