Ex. no: 9

Name : Ashish P Shaji

Roll NO : 230701041

DEADLOCK AVOIDANCE

Aim:

To find out a safe sequence using Banker's algorithm for deadlock avoidance.

Algorithm:

1. Initialize work=available and finish[i]=false for all values of i

2. Find an i such that both:

finish[i]=false and Needi<= work

3. If no such i exists go to step 6

4. Compute work=work+allocationi

5. Assign finish[i] to true and go to step 2

6. If finish[i]==true for all i, then print safe sequence

7. Else print there is no safe sequence

Program Code:

```c
#include <stdio.h>
#include <stdbool.h>

#define P 5 // Number of processes
#define R 3 // Number of resource types

bool isSafe(int processes[], int available[], int max[][R], int allocation[][R]) {
    int need[P][R];
    for (int i = 0; i < P; i++) {
        for (int j = 0; j < R; j++) {
            need[i][j] = max[i][j] - allocation[i][j];
        }
    }

    bool finish[P] = {false};
    int safeSequence[P];
    int work[R];

    for (int i = 0; i < R; i++) {
        work[i] = available[i];
    }

    int count = 0;
    while (count < P) {
        bool found = false;
        for (int p = 0; p < P; p++) {
            if (!finish[p]) {
                bool canProceed = true;
                for (int j = 0; j < R; j++) {
                    if (need[p][j] > work[j]) {
                        canProceed = false;
                        break;
                    }
                }

                if (canProceed) {
                    for (int k = 0; k < R; k++) {
                        work[k] += allocation[p][k];
                    }
                    safeSequence[count++] = p;
                    finish[p] = true;
                    found = true;
                }
            }
        }

        if (!found) {
            printf("No safe sequence found.\n");
            return false;
        }
    }
```

```
    printf("Safe sequence is: ");
    for (int i = 0; i < P; i++) {
        printf("P%d ", safeSequence[i]);
    }
    printf("\n");
    return true;
}

int main() {
    int processes[] = {0, 1, 2, 3, 4};
    int available[] = {3, 3, 2};
    int max[P][R] = {
        {7, 5, 3},
        {3, 2, 2},
        {9, 0, 2},
        {2, 2, 2},
        {4, 3, 3}
    };
    int allocation[P][R] = {
        {0, 1, 0},
        {2, 0, 0},
        {3, 0, 2},
        {2, 1, 1},
        {0, 0, 2}
    };

    isSafe(processes, available, max, allocation);
    return 0;
}
```

OUTPUT :

```
The SAFE Sequence is: P1 -> P3 -> P4 -> P0 -> P2
```