

Ex. No.: 9

NAME: T.R.DIVYASREE

Roll No. :230701083

DEADLOCK AVOIDANCE

Aim:

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

Program Code:

```
#include <stdio.h>

#include <stdlib.h>

#define P 5 // Number of
processes #define R 3 // Number
of resources

int main() {
    int need[P][R], allot[P][R], max[P][R],
    avail[R]; int finish[P] = {0}, safeSeq[P];
    int i, j, k;
    // Example values (you can modify or take
    as input) int allocation[P][R] = {
        {0, 1, 0},
        {2, 0, 0},
        {3, 0, 2},
        {2, 1, 1},
        {0, 0, 2}
    };
    int maximum[P][R] = {
        {7, 5, 3},
        {3, 2, 2},
        {9, 0, 2},
        {2, 2, 2},
        {4, 3, 3}
    };
    int available[R] = {3, 3, 2};

    // Calculate need
    matrix for (i = 0; i <
P; i++) {
        for (j = 0; j < R; j++) {
            need[i][j] = maximum[i][j] -
allocation[i][j]; allot[i][j] =
allocation[i][j];
        }
    }
}
```

```
int count
```

```
= 0; int  
work[R];
```

```

        for (i = 0; i < R;
i++) work[i] =
available[i];
        while (count <
P) { bool found =
false;
            for (i = 0; i
< P; i++) { if (!
finish[i]) {
                for (j = 0; j < R; j++)
                    if (need[i]
[j] > work[j]) break;

                if (j == R) {
                    for (k =
0; k < R; k++)
                        work[k] += allot[i]
[k];

                    safeSeq[co
unt++] = i; finish[i] =
1;

                    found = true;
                }
            }
        }

        if (!found) {
            printf("\nNo SAFE Sequence Found (System is in UNSAFE
state)\n"); return 0;
        }
    }

    // Print the Safe Sequence

    printf("The SAFE Sequence is:
\n"); for (i = 0; i < P; i++)
        printf("P%d%s", safeSeq[i], (i < P - 1) ? " -> " : "\n");

    return 0;
}

```

OUTPUT :

```

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

```