**Ex. No.: 9**

**Date:** 22/2/25

**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:**

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

#define MAX 10

void findSafeSequence(int n, int m, int available[], int max[][MAX], int allocation[][MAX]) { int work[MAX], finish[MAX] = {0}, safeSeq[MAX], need[MAX][MAX];

for (int i = 0; i < m; i++) work[i] = available[i]; for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

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

int count = 0; while (count < n) {

```
bool found = false;
for (int i = 0; i < n; i++) { if (!finish[i]) {
bool canAllocate = true; for (int j = 0; j < m; j++)
if (need[i][j] > work[j]) { canAllocate = false; break; } if (canAllocate) {
for (int j = 0; j < m; j++) work[j] += allocation[i][j]; safeSeq[count++] = i;
finish[i] = 1; found = true;
}
}
}
```

```
if (!found) { printf("No safe sequence.\n"); return; }
}
printf("Safe sequence: ");
for (int i = 0; i < n; i++) printf("P%d ", safeSeq[i]); printf("\n");
}
int main() {
int n, m, available[MAX], max[MAX][MAX], allocation[MAX][MAX];
printf("Enter processes and resources: "); scanf("%d %d", &n, &m);
while (getchar() != '\n');
printf("Enter available resources: ");
for (int i = 0; i < m; i++) scanf("%d", &available[i]); while (getchar() != '\n');

printf("Enter Max matrix: \n"); for (int i = 0; i < n; i++)
for (int j = 0; j < m; j++) scanf("%d", &max[i][j]); while (getchar() != '\n');
printf("Enter Allocation matrix: \n"); for (int i = 0; i < n; i++)
for (int j = 0; j < m; j++) scanf("%d", &allocation[i][j]); while (getchar() != '\n');
findSafeSequence(n, m, available, max, allocation); return 0;
}
```


**RESULT:**

The Safe Sequence is found using Banker's Algorithm for Deadlock Avoidance.