Program-5

Write a C program to simulate: a) Bankers' algorithm for the purpose of deadlock avoidance.

Code:

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

#define MAX_PROCESSES 10
#define MAX_RESOURCES 10

int main() {
    int n, m;
    int alloc[MAX_PROCESSES][MAX_RESOURCES];
    int max[MAX_PROCESSES][MAX_RESOURCES];
    int avail[MAX_RESOURCES];
    int need[MAX_PROCESSES][MAX_RESOURCES];
    bool finished[MAX_PROCESSES] = {false};
    int safe_sequence[MAX_PROCESSES];
    int count = 0;

    printf("Enter number of processes and resources: ");
    scanf("%d %d", &n, &m);

    printf("Enter allocation matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &alloc[i][j]);
        }
    }

    printf("Enter max matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    printf("Enter available matrix:\n");
```

```
for (int j = 0; j < m; j++) {
   scanf("%d", &avail[j]);
}

for (int i = 0; i < n; i++) {
   for (int j = 0; j < m; j++) {
      need[i][j] = max[i][j] - alloc[i][j];
   }
}
printf("\nProcess\t\tAllocation\tMax\t\tNeed\n");
for (int i = 0; i < n; i++) {
   printf("P%d\t\t", i);

   for (int j = 0; j < m; j++) {
      printf("%d ", alloc[i][j]);
   }
   printf("\t");

   for (int j = 0; j < m; j++) {
      printf("%d ", max[i][j]);
   }
   printf("\t");

   for (int j = 0; j < m; j++) {
      printf("%d ", need[i][j]);
   }
   printf("\n");
}

while (count < n) {
   bool found = false;
   for (int i = 0; i < n; i++) {
      if (!finished[i]) {
         int j;
         for (j = 0; j < m; j++) {
            if (need[i][j] > avail[j]) {
               break;
            }
         }
```

```
            if (j == m) {
                for (int k = 0; k < m; k++) {
                    avail[k] += alloc[i][k];
                }
                safe_sequence[count++] = i;
                finished[i] = true;
                found = true;
            }
        }
    }

    if (!found) {
        printf("System is not in safe state.\n");
        return 0;
    }
  }

  printf("System is in safe state.\n");
  printf("Safe sequence is: ");
  for (int i = 0; i < n; i++) {
    printf("P%d", safe_sequence[i]);
    if (i != n - 1) {
        printf(" -> ");
    }
  }
  printf("\n");

  return 0;
}
```

Output:

```
Enter number of processes and resources: 5 3
Enter allocation matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter max matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter available matrix:
3 3 2

Process         Allocation      Max             Need
P0              0 1 0   7 5 3   7 4 3
P1              2 0 0   3 2 2   1 2 2
P2              3 0 2   9 0 2   6 0 0
P3              2 1 1   2 2 2   0 1 1
P4              0 0 2   4 3 3   4 3 1
System is in safe state.
Safe sequence is: P1 -> P3 -> P4 -> P0 -> P2

Process returned 0 (0x0)   execution time : 77.718 s
Press any key to continue.
```