EXP 17: Illustrate the deadlock avoidance concept by simulating Banker's algorithm with C.

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

#define MAX 10

int main() {
  int n, m; // n = processes, m = resources
  int alloc[MAX][MAX], max[MAX][MAX], need[MAX][MAX], avail[MAX];
  int finish[MAX] = {0}, safeSeq[MAX];
  int i, j, k;

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

  // Input Allocation Matrix
  printf("Enter Allocation Matrix (%d x %d):\n", n, m);
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      scanf("%d", &alloc[i][j]);

  // Input Max Matrix
  printf("Enter Maximum Matrix (%d x %d):\n", n, m);
```

```
for (i = 0; i < n; i++)
   for (j = 0; j < m; j++)
      scanf("%d", &max[i][j]);

// Input Available Resources
printf("Enter Available Resources (%d):\n", m);
for (i = 0; i < m; i++)
   scanf("%d", &avail[i]);

// Calculate Need Matrix = Max - Allocation
for (i = 0; i < n; i++)
   for (j = 0; j < m; j++)
      need[i][j] = max[i][j] - alloc[i][j];

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

         if (canAllocate) {
```

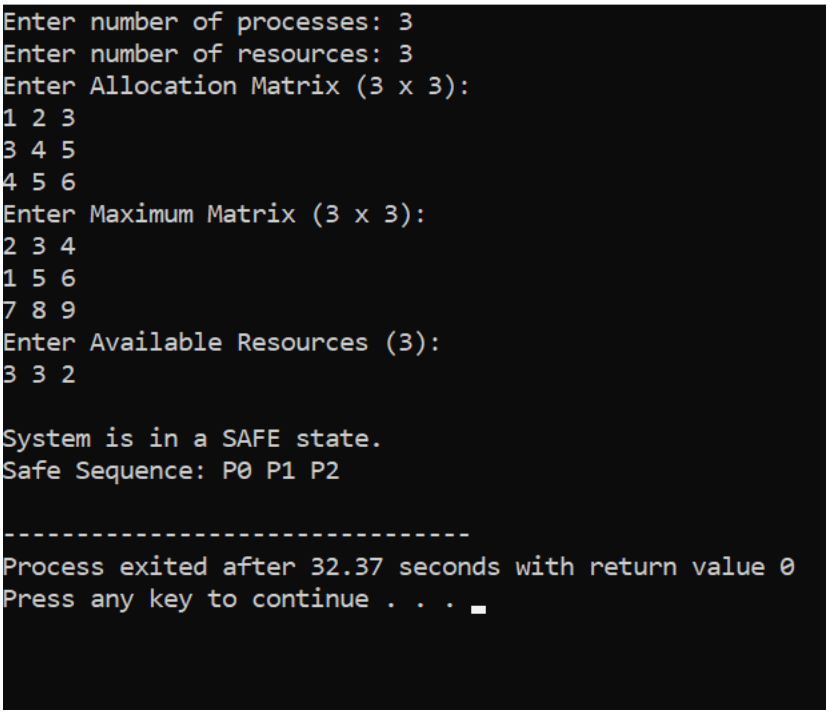
```
            // Add allocated resources back to available
            for (k = 0; k < m; k++)
               avail[k] += alloc[i][k];
            safeSeq[count++] = i;
            finish[i] = 1;
            found = true;
          }
        }
      }

      if (!found) {
        printf("\nSystem is NOT in a safe state (deadlock may occur).\n");
        return 1;
      }
    }

    // If system is in safe state
    printf("\nSystem is in a SAFE state.\nSafe Sequence: ");
    for (i = 0; i < n; i++)
      printf("P%d ", safeSeq[i]);
    printf("\n");

    return 0;
}
```

Sample Output


Enter number of processes: 3
Enter number of resources: 3
Enter Allocation Matrix (3 x 3):
1 2 3
3 4 5
4 5 6
Enter Maximum Matrix (3 x 3):
2 3 4
1 5 6
7 8 9
Enter Available Resources (3):
3 3 2
System is in a SAFE state.
Safe Sequence: P0 P1 P2

Process exited after 32.37 seconds with return value 0
Press any key to continue . . .