Program 5. Develop a C program to simulate Bankers Algorithm for DeadLock Avoidance.

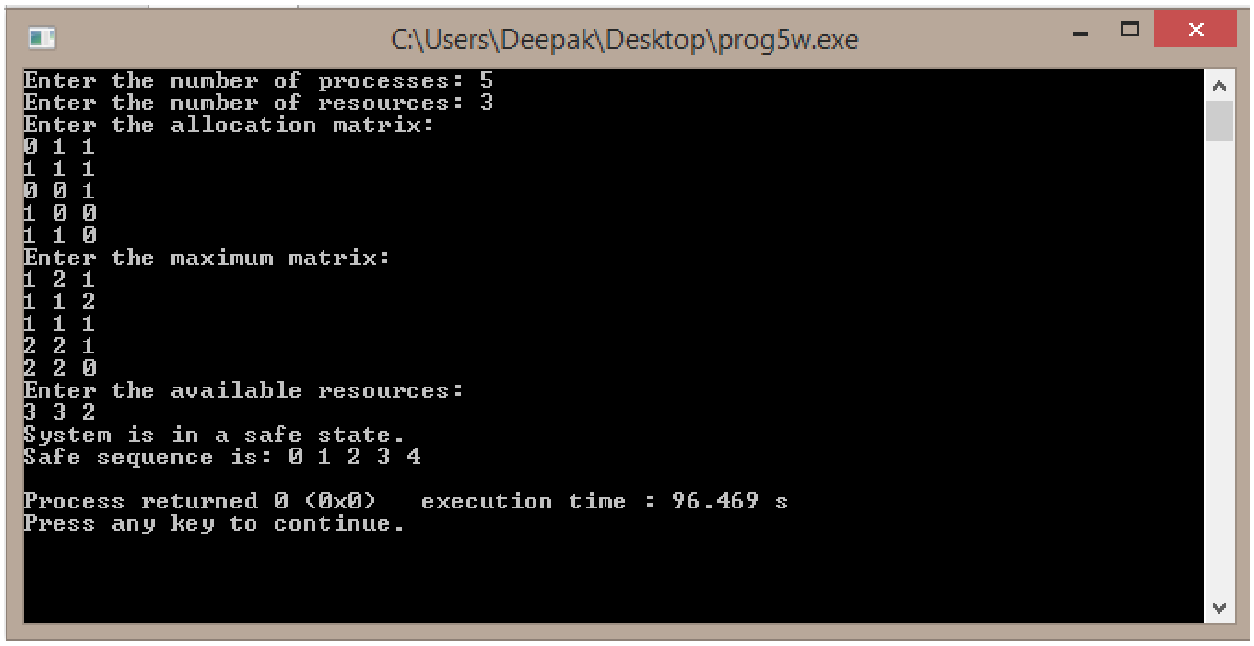
```
#include <stdio.h>
#define MAX_PROCESS 10
#define MAX_RESOURCE 10
int main() {
  int n, m, i, j, k;
  int allocation[MAX_PROCESS][MAX_RESOURCE],
    max[MAX_PROCESS][MAX_RESOURCE], available[MAX_RESOURCE];
  int need[MAX_PROCESS][MAX_RESOURCE], finish[MAX_PROCESS],
    safeSeq[MAX_PROCESS], work[MAX_RESOURCE];
  // Input number of processes and resources
  printf("Enter the number of processes: ");
  scanf("%d", &n);
  printf("Enter the number of resources: ");
  scanf("%d", &m);
  // Input the allocation matrix
  printf("Enter the allocation matrix:\n");
  for (i = 0; i < n; ++i) {
    for (j = 0; j < m; ++j) {
      scanf("%d", &allocation[i][j]);
    }
  }


  // Input the maximum matrix
  printf("Enter the maximum matrix:\n");
```

```
for (i = 0; i < n; ++i) {
    for (j = 0; j < m; ++j) {
        scanf("%d", &max[i][j]);
    }
}
// Input the available resources
printf("Enter the available resources:\n");
for (i = 0; i < m; ++i) {
    scanf("%d", &available[i]);
}
// Initialize finish array
for (i = 0; i < n; ++i) {
    finish[i] = 0;
}
// Calculate need matrix
for (i = 0; i < n; ++i) {
    for (j = 0; j < m; ++j) {
        need[i][j] = max[i][j] - allocation[i][j];
    }
}
// Initialize work array with available resources
for (i = 0; i < m; ++i) {
    work[i] = available[i];
}
int count = 0;
while (count < n) {
    int found = 0;
    for (i = 0; i < n; ++i) {
        if (finish[i] == 0) {
            int flag = 1;
            for (j = 0; j < m; ++j) {
                if (need[i][j] > work[j]) {
                    flag = 0;
                    break;
                }
            }
            if (flag) {
                // Add allocation to work
                for (k = 0; k < m; ++k) {
                    work[k] += allocation[i][k];
                }
                // Record the safe sequence
                safeSeq[count++] = i;
                finish[i] = 1;
                found = 1;
            }
        }
    }
    if (!found) {
        printf("System is not in a safe state!\n");
        return 0;
    }
}
// Print the safe sequence
printf("System is in a safe state.\n");
printf("Safe sequence is: ");
```

```
    for (i = 0; i < n; ++i) {
        printf("%d ", safeSeq[i]);
    }
    printf("\n");
    return 0;
}
```

Output



C:\Users\Deepak\Desktop\prog5w.exe

```
Enter the number of processes: 5
Enter the number of resources: 3
Enter the allocation matrix:
0 1 1
1 1 1
0 0 1
1 0 0
1 1 0
Enter the maximum matrix:
1 2 1
1 1 2
1 1 1
2 2 1
2 2 0
Enter the available resources:
3 3 2
System is in a safe state.
Safe sequence is: 0 1 2 3 4

Process returned 0 (0x0)   execution time : 96.469 s
Press any key to continue.
```