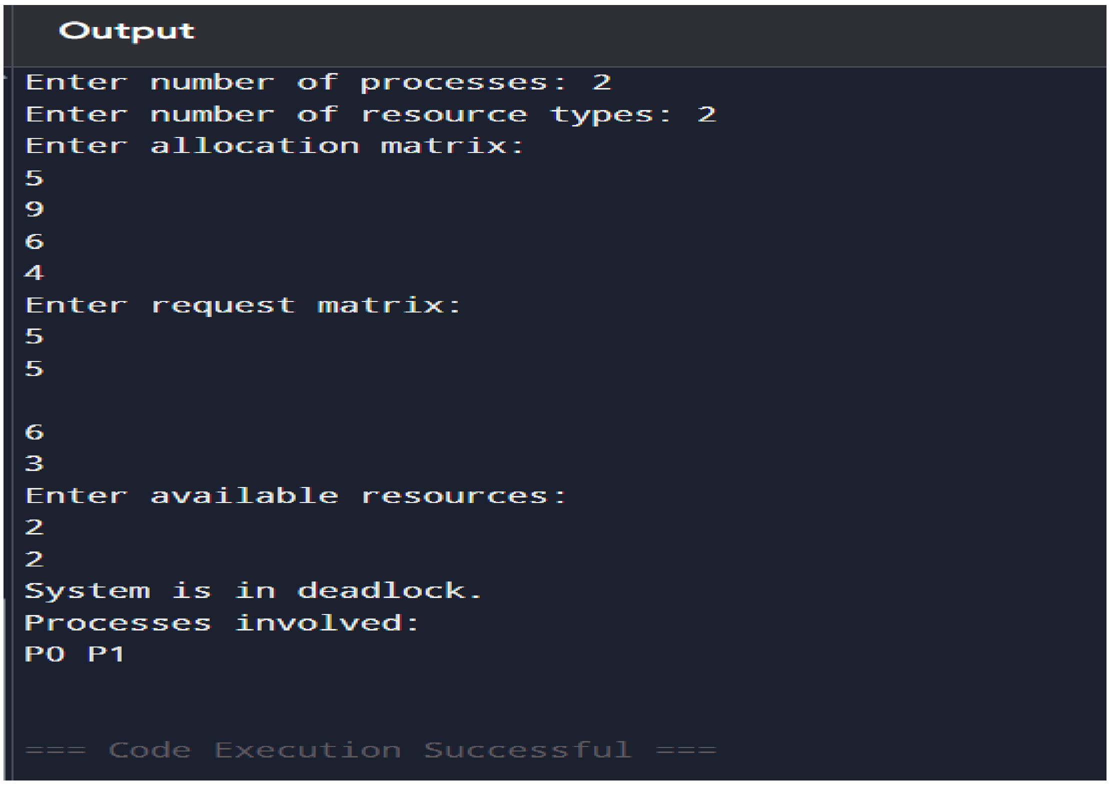
b) Deadlock Detection

```c
#include <stdio.h>

int main() {
  int n, m, i, j, k;

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

  printf("Enter number of resource types: ");
  scanf("%d", &m);

  int alloc[n][m], req[n][m], avail[m], finish[n];

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

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

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

  for(i = 0; i < n; i++)
    finish[i] = 0;

  int changed = 1;
  while(changed) {
    changed = 0;
    for(i = 0; i < n; i++) {
      if(finish[i] == 0) {
        int canFinish = 1;
        for(j = 0; j < m; j++) {
          if(req[i][j] > avail[j]) {
            canFinish = 0;
            break;
          }
        }
        if(canFinish) {
          for(k = 0; k < m; k++)
            avail[k] += alloc[i][k];
          finish[i] = 1;
```

```
                changed = 1;
            }
        }
    }
}

int deadlocked = 0;
for(i = 0; i < n; i++) {
    if(finish[i] == 0) {
        deadlocked = 1;
        break;
    }
}

if(deadlocked) {
    printf("System is in deadlock.\nProcesses involved:\n");
    for(i = 0; i < n; i++) {
        if(finish[i] == 0)
            printf("P%d ", i);
    }
    printf("\n");
} else {
    printf("System is not in deadlock.\n");
}

return 0;
}
```



```
Output
Enter number of processes: 2
Enter number of resource types: 2
Enter allocation matrix:
5
9
6
4
Enter request matrix:
5
5

6
3
Enter available resources:
2
2
System is in deadlock.
Processes involved:
P0 P1


=== Code Execution Successful ===
```