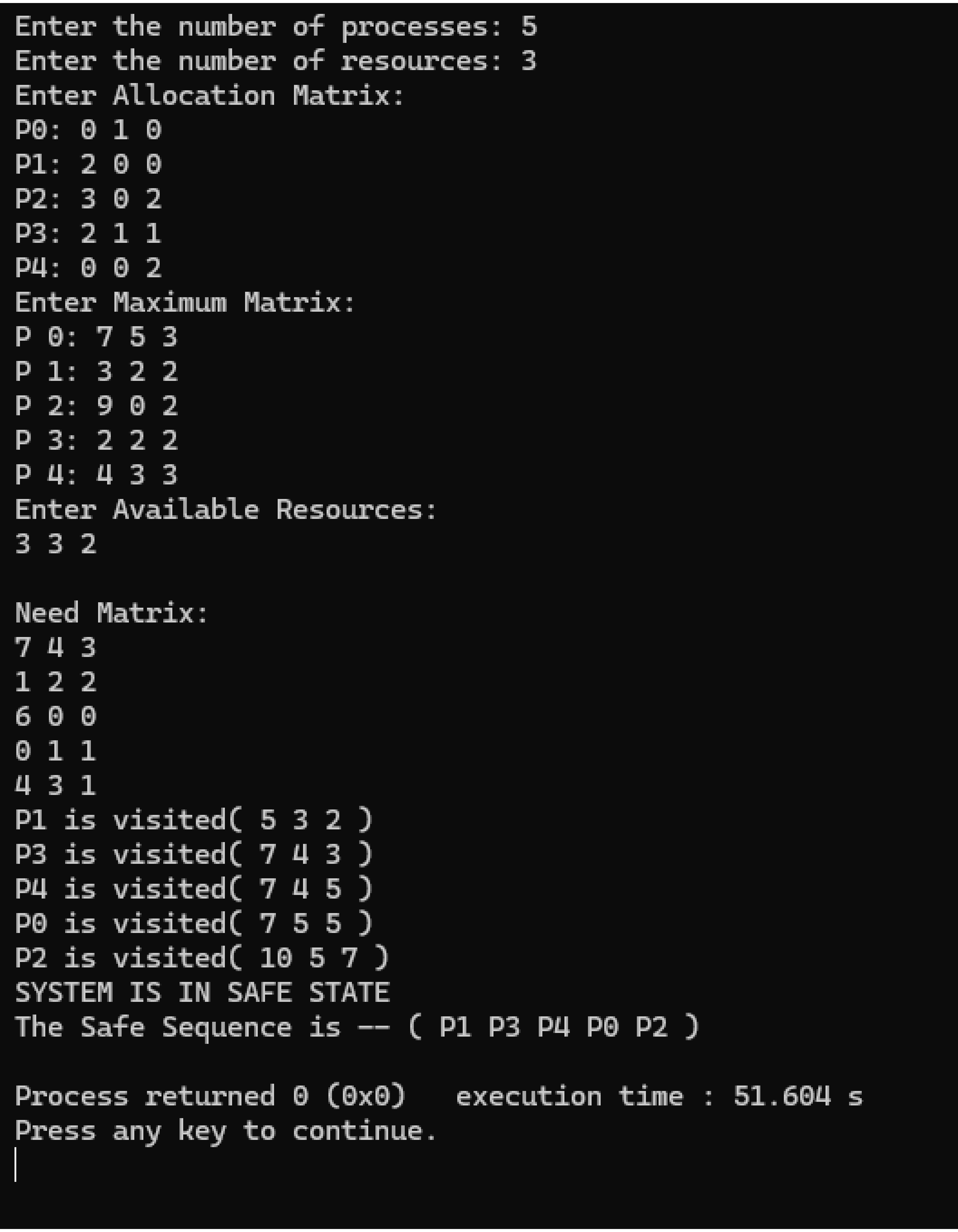
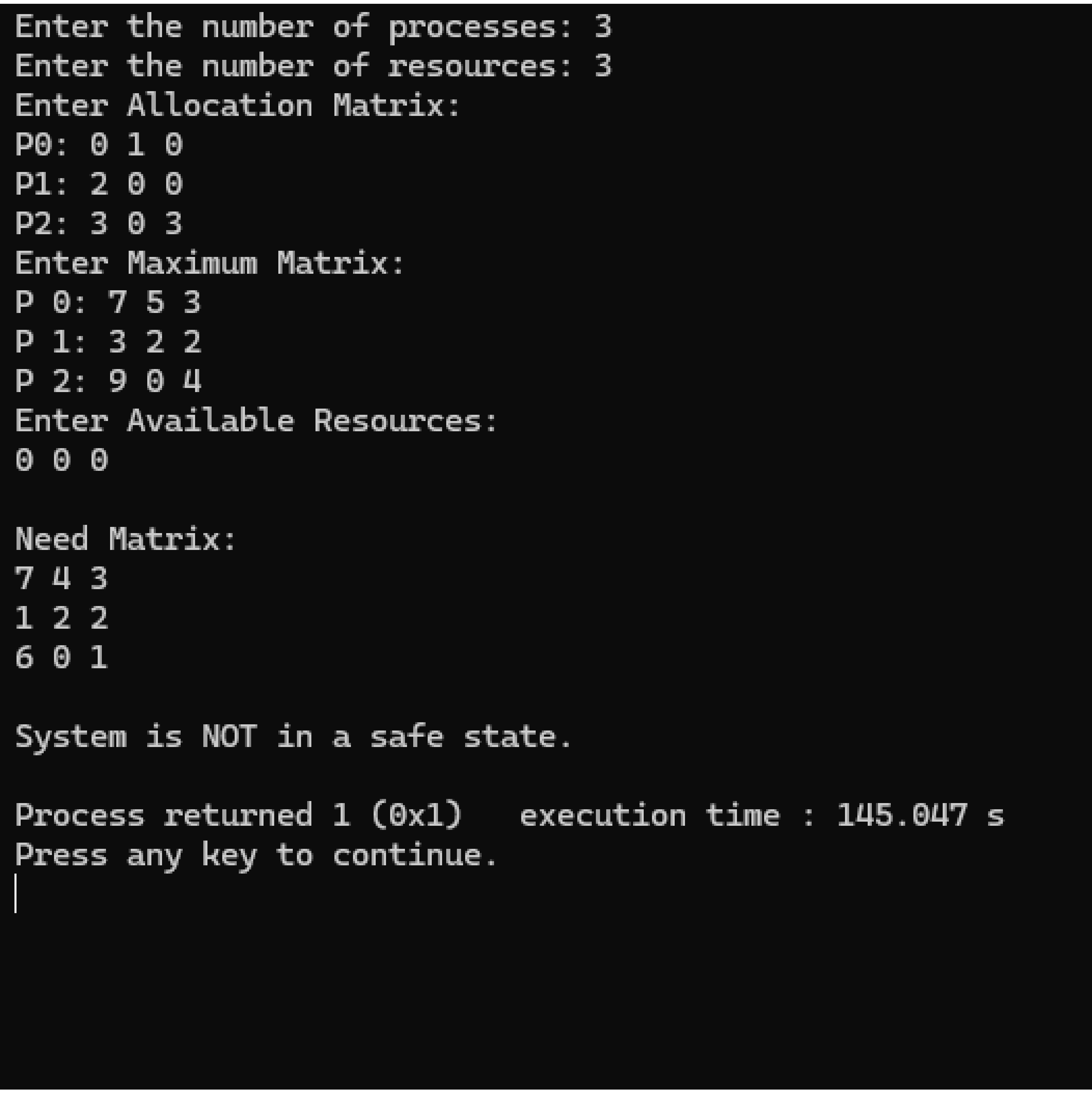
```c
#include <stdio.h>
#include <stdbool.h>
int main() {
    int P, R;
    printf("Enter the number of processes: ");
    scanf("%d", &P);
    printf("Enter the number of resources: ");
    scanf("%d", &R);

    int alloc[P][R], max[P][R], need[P][R], avail[R];
    int i, j;
    printf("Enter Allocation Matrix:\n");
    for (i = 0; i < P; i++) {
        printf("P%d: ", i);
        for (j = 0; j < R; j++) {
            scanf("%d", &alloc[i][j]);
        }}
    printf("Enter Maximum Matrix:\n");
    for (i = 0; i < P; i++) {
        printf("P %d: ", i);
        for (j = 0; j < R; j++) {
            scanf("%d", &max[i][j]);
        }}
    printf("Enter Available Resources:\n");
    for (i = 0; i < R; i++) {
        scanf("%d", &avail[i]);
    }
    printf("\nNeed Matrix:\n");
    for (i = 0; i < P; i++) {
        for (j = 0; j < R; j++) {
            need[i][j] = max[i][j] - alloc[i][j];
            printf("%d ", need[i][j]);
        }
        printf("\n");
    }
    bool finish[P];
    int safeSeq[P], work[R];
    for (i = 0; i < P; i++) finish[i] = false;
    for (i = 0; i < R; i++) work[i] = avail[i];
    int count = 0;
```

```
        printf("\n");
    }
    bool finish[P];
    int safeSeq[P], work[R];
    for (i = 0; i < P; i++) finish[i] = false;
    for (i = 0; i < R; i++) work[i] = avail[i];
    int count = 0;
    while (count < P) {
        bool found = false;
        for (i = 0; i < P; i++) {
            if (!finish[i]) {
                bool canAllocate = true;
                for (j = 0; j < R; j++) {
                    if (need[i][j] > work[j]) {
                        canAllocate = false;
                        break;
                    }}
           if (canAllocate) {
                    printf("P%d is visited( ", i);
                    for (j = 0; j < R; j++) {
                        work[j] += alloc[i][j];
                        printf("%d ", work[j]);
                    }
                    printf(")\n");

                    safeSeq[count++] = i;
                    finish[i] = true;
                    found = true;
                }}}
          if (!found) {
            printf("\nSystem is NOT in a safe state.\n");
            return 1;
        }}
printf("SYSTEM IS IN SAFE STATE\nThe Safe Sequence is -- ( ");
    for (i = 0; i < P; i++) {
        printf("P%d ", safeSeq[i]);}
    printf(")\n");
    return 0;
}
```



```
Enter the number of processes: 5
Enter the number of resources: 3
Enter Allocation Matrix:
P0: 0 1 0
P1: 2 0 0
P2: 3 0 2
P3: 2 1 1
P4: 0 0 2
Enter Maximum Matrix:
P 0: 7 5 3
P 1: 3 2 2
P 2: 9 0 2
P 3: 2 2 2
P 4: 4 3 3
Enter Available Resources:
3 3 2

Need Matrix:
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1
P1 is visited( 5 3 2 )
P3 is visited( 7 4 3 )
P4 is visited( 7 4 5 )
P0 is visited( 7 5 5 )
P2 is visited( 10 5 7 )
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- ( P1 P3 P4 P0 P2 )

Process returned 0 (0x0)   execution time : 51.604 s
Press any key to continue.
```


```
Enter the number of processes: 3
Enter the number of resources: 3
Enter Allocation Matrix:
P0: 0 1 0
P1: 2 0 0
P2: 3 0 3
Enter Maximum Matrix:
P 0: 7 5 3
P 1: 3 2 2
P 2: 9 0 4
Enter Available Resources:
0 0 0

Need Matrix:
7 4 3
1 2 2
6 0 1

System is NOT in a safe state.

Process returned 1 (0x1)   execution time : 145.047 s
Press any key to continue.
```