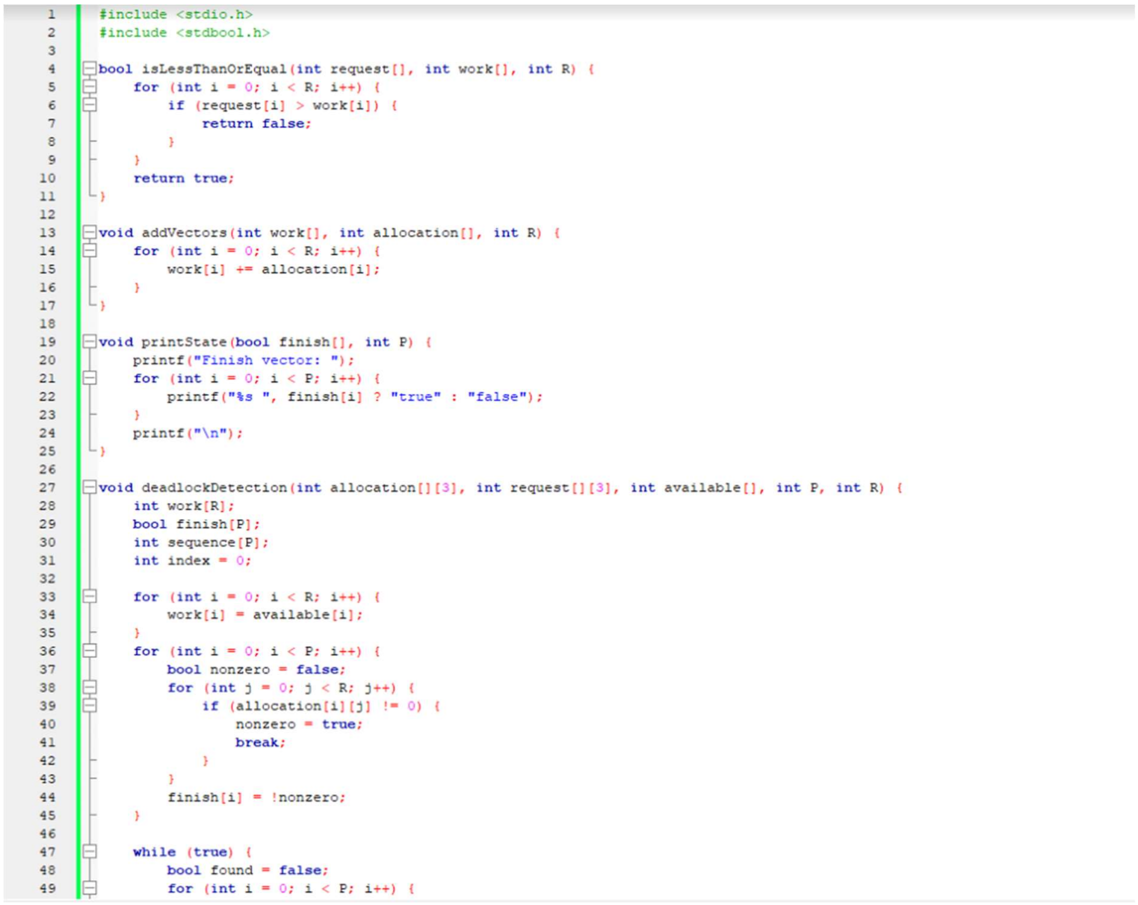
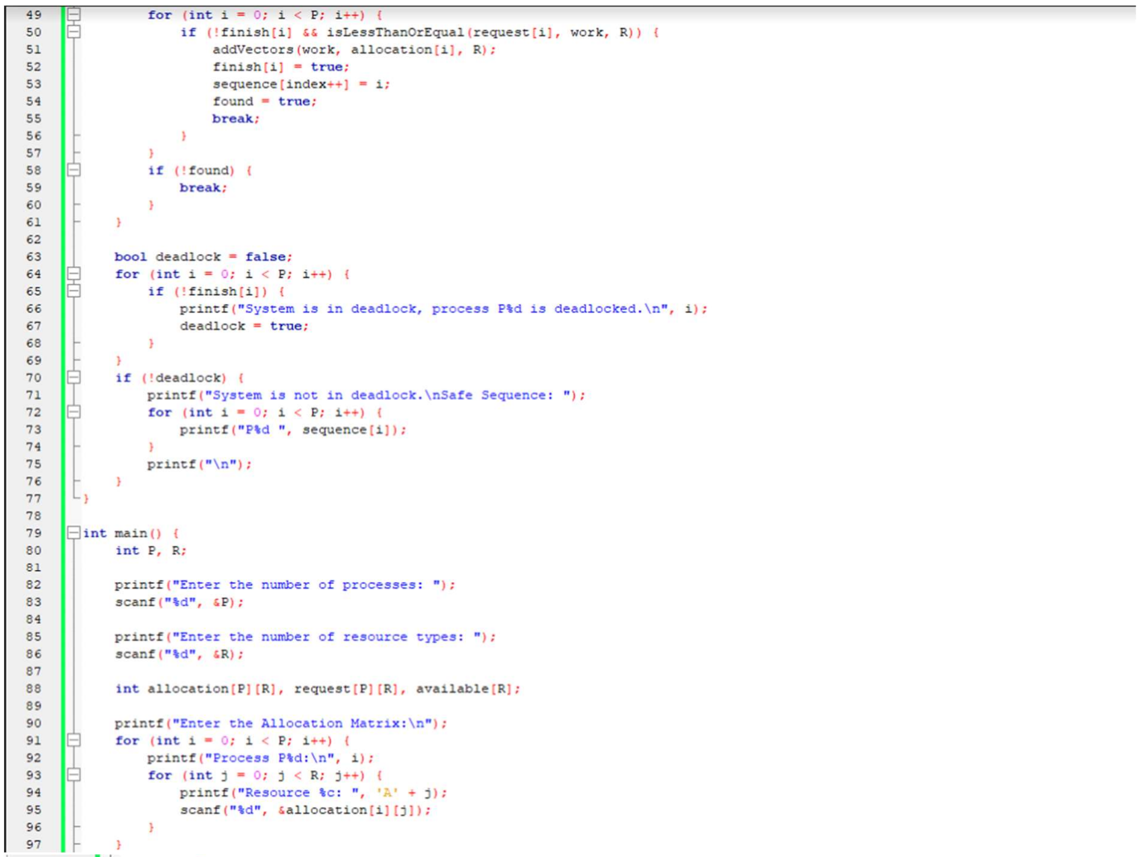
LAB08

Write a C program to simulate deadlock detection

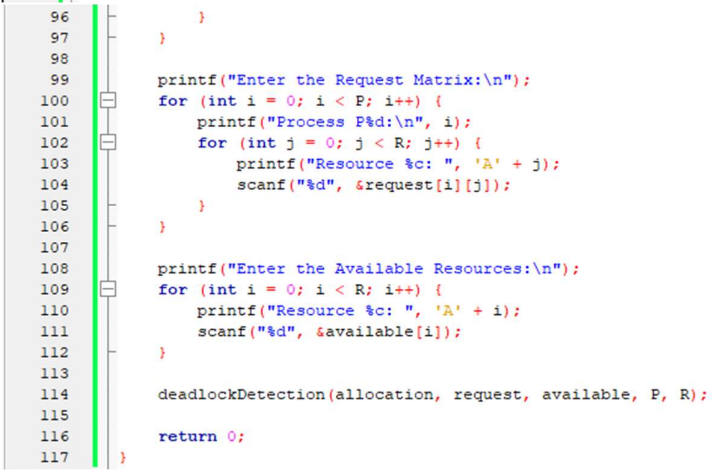
INPUT:



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

bool isLessThanOrEqual(int request[], int work[], int R) {
    for (int i = 0; i < R; i++) {
        if (request[i] > work[i]) {
            return false;
        }
    }
    return true;
}

void addVectors(int work[], int allocation[], int R) {
    for (int i = 0; i < R; i++) {
        work[i] += allocation[i];
    }
}

void printState(bool finish[], int P) {
    printf("Finish vector: ");
    for (int i = 0; i < P; i++) {
        printf("%s ", finish[i] ? "true" : "false");
    }
    printf("\n");
}

void deadlockDetection(int allocation[][3], int request[][3], int available[], int P, int R) {
    int work[R];
    bool finish[P];
    int sequence[P];
    int index = 0;

    for (int i = 0; i < R; i++) {
        work[i] = available[i];
    }
    for (int i = 0; i < P; i++) {
        bool nonzero = false;
        for (int j = 0; j < R; j++) {
            if (allocation[i][j] != 0) {
                nonzero = true;
                break;
            }
        }
        finish[i] = !nonzero;
    }

    while (true) {
        bool found = false;
        for (int i = 0; i < P; i++) {
```



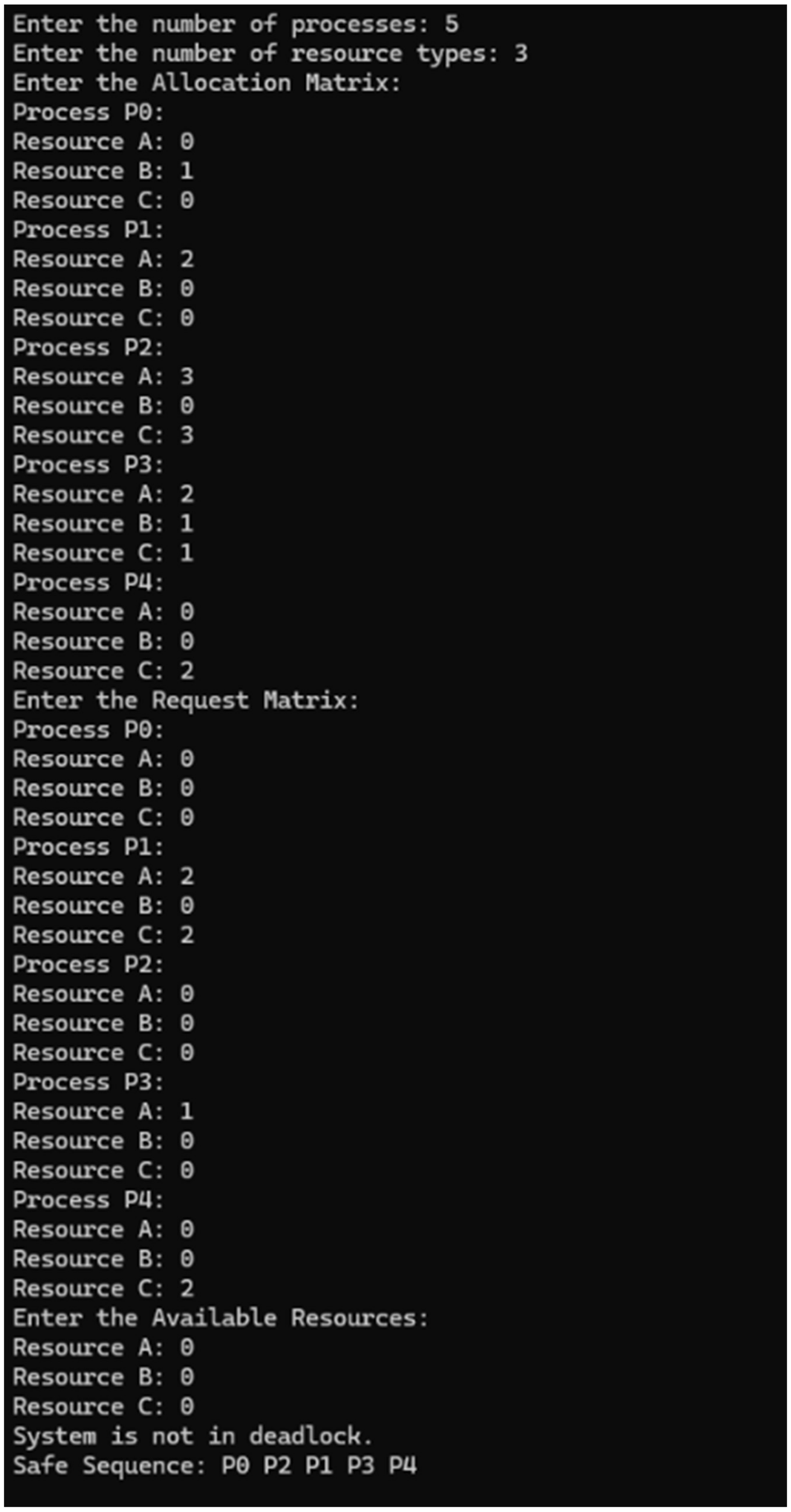
```c
        for (int i = 0; i < P; i++) {
            if (!finish[i] && isLessThanOrEqual(request[i], work, R)) {
                addVectors(work, allocation[i], R);
                finish[i] = true;
                sequence[index++] = i;
                found = true;
                break;
            }
        }
        if (!found) {
            break;
        }
    }

    bool deadlock = false;
    for (int i = 0; i < P; i++) {
        if (!finish[i]) {
            printf("System is in deadlock, process P%d is deadlocked.\n", i);
            deadlock = true;
        }
    }
    if (!deadlock) {
        printf("System is not in deadlock.\nSafe Sequence: ");
        for (int i = 0; i < P; i++) {
            printf("P%d ", sequence[i]);
        }
        printf("\n");
    }
}

int main() {
    int P, R;

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

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

    int allocation[P][R], request[P][R], available[R];

    printf("Enter the Allocation Matrix:\n");
    for (int i = 0; i < P; i++) {
        printf("Process P%d:\n", i);
        for (int j = 0; j < R; j++) {
            printf("Resource %c: ", 'A' + j);
            scanf("%d", &allocation[i][j]);
        }
    }
```



```c
        }
    }
```

```c

    printf("Enter the Request Matrix:\n");
    for (int i = 0; i < P; i++) {
        printf("Process P%d:\n", i);
        for (int j = 0; j < R; j++) {
            printf("Resource %c: ", 'A' + j);
            scanf("%d", &request[i][j]);
        }
    }

    printf("Enter the Available Resources:\n");
    for (int i = 0; i < R; i++) {
        printf("Resource %c: ", 'A' + i);
        scanf("%d", &available[i]);
    }

    deadlockDetection(allocation, request, available, P, R);

    return 0;
}
```

OUTPUT:



```
Enter the number of processes: 5
Enter the number of resource types: 3
Enter the Allocation Matrix:
Process P0:
Resource A: 0
Resource B: 1
Resource C: 0
Process P1:
Resource A: 2
Resource B: 0
Resource C: 0
Process P2:
Resource A: 3
Resource B: 0
Resource C: 3
Process P3:
Resource A: 2
Resource B: 1
Resource C: 1
Process P4:
Resource A: 0
Resource B: 0
Resource C: 2
Enter the Request Matrix:
Process P0:
Resource A: 0
Resource B: 0
Resource C: 0
Process P1:
Resource A: 2
Resource B: 0
Resource C: 2
Process P2:
Resource A: 0
Resource B: 0
Resource C: 0
Process P3:
Resource A: 1
Resource B: 0
Resource C: 0
Process P4:
Resource A: 0
Resource B: 0
Resource C: 2
Enter the Available Resources:
Resource A: 0
Resource B: 0
Resource C: 0
System is not in deadlock.
Safe Sequence: P0 P2 P1 P3 P4
```