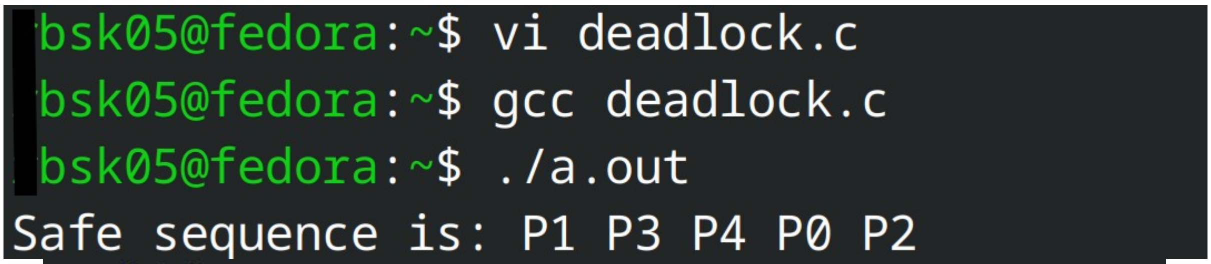
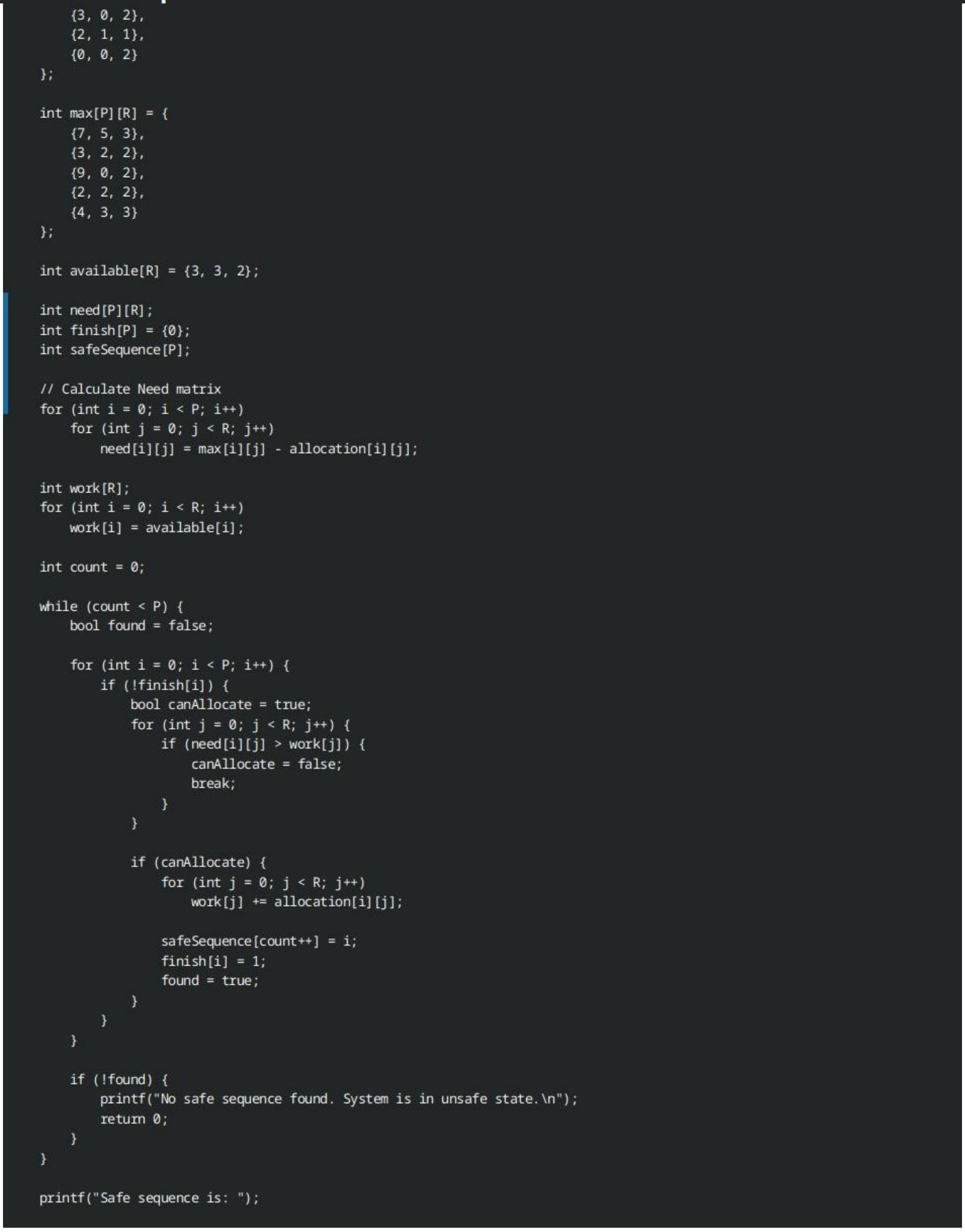
DEADLOCK AVOIDANCE

AADITYA P

230701001



```
[redacted]bsk05@fedora:~$ vi deadlock.c
bsk05@fedora:~$ gcc deadlock.c
bsk05@fedora:~$ ./a.out
Safe sequence is: P1 P3 P4 P0 P2
```



```
        {3, 0, 2},
        {2, 1, 1},
        {0, 0, 2}
    };

    int max[P][R] = {
        {7, 5, 3},
        {3, 2, 2},
        {9, 0, 2},
        {2, 2, 2},
        {4, 3, 3}
    };

    int available[R] = {3, 3, 2};

    int need[P][R];
    int finish[P] = {0};
    int safeSequence[P];

    // Calculate Need matrix
    for (int i = 0; i < P; i++)
        for (int j = 0; j < R; j++)
            need[i][j] = max[i][j] - allocation[i][j];

    int work[R];
    for (int i = 0; i < R; i++)
        work[i] = available[i];

    int count = 0;

    while (count < P) {
        bool found = false;

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

                if (canAllocate) {
                    for (int j = 0; j < R; j++)
                        work[j] += allocation[i][j];

                    safeSequence[count++] = i;
                    finish[i] = 1;
                    found = true;
                }
            }
        }

        if (!found) {
            printf("No safe sequence found. System is in unsafe state.\n");
            return 0;
        }
    }

    printf("Safe sequence is: ");
```