Kamalesh S P
230701138
II-CSE-B

9) Banker's algorithm for deadlock avoidance

Program Code:

```
    int avail[R] = {3, 3, 2};

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

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

    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] > avail[j]) {
                        canAllocate = false;
                        break;
                    }
                }

                if (canAllocate) {
                    for (j = 0; j < R; j++)
                        avail[j] += alloc[i][j];

                    safeSeq[count++] = i;
                    finish[i] = true;
                    found = true;
                }
            }
        }

        if (!found) {
            printf("System is not in a safe state.\n");
            return 1;
        }
    }

    printf("System is in a safe state.\nSafe sequence is: ");
    for (i = 0; i < P; i++)
        printf("P%d ", safeSeq[i]);
    printf("\n");

    return 0;
}
```

Output:

```
System is in a safe state.
Safe sequence is: P1 P3 P4 P0 P2
```