Assignment 5

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

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

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

    int allocation[n][m];  // Allocation matrix
    int max[n][m];         // Maximum demand matrix
    int available[m];      // Available resources array
    int need[n][m];        // Need matrix
    int safeSequence[n];   // Safe sequence array

    bool finish[n];        // Array to mark finished processes

    printf("Enter the Allocation Matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            scanf("%d", &allocation[i][j]);
        }
    }

    printf("Enter the Maximum Matrix:\n");
    for (i = 0; i < n; i++) {
```

```
        for (j = 0; j < m; j++) {
            scanf("%d", &max[i][j]);
        }
    }

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

    // Calculate the Need matrix
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            need[i][j] = max[i][j] - allocation[i][j];
        }
    }

    printf("\nNeed Matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            printf("%d ", need[i][j]);
        }
        printf("\n");
    }

    int count = 0;
    while (count < n) {
        bool found = false;
        for (i = 0; i < n; i++) {
            if (finish[i] == false) {  // Check if the process is
not yet finished
                bool can_allocate = true;
                for (j = 0; j < m; j++) {
```

```
                    if (need[i][j] > available[j]) {
                        can_allocate = false;
                        break;
                    }
                }

                if (can_allocate) {
                    for (k = 0; k < m; k++) {
                        available[k] += allocation[i][k];
                    }
                    safeSequence[count++] = i;
                    finish[i] = true;
                    found = true;
                    printf("Process %d has been allocated resources
and finished.\n", i);
                }
            }
        }

        if (found == false) {
            printf("System is in an unsafe state. Deadlock may
occur.\n");
            return 0;
        }
    }

    // If we reach here, it means system is in safe state
    printf("System is in a safe state.\nSafe Sequence: ");
    for (i = 0; i < n; i++) {
        printf("%d ", safeSequence[i]);
    }
    printf("\n");
```

```
    return 0;
}
```

Output :

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

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


Process 1 has been allocated resources and finished.
Process 3 has been allocated resources and finished.
```

```
Process 4 has been allocated resources and finished.
Process 0 has been allocated resources and finished.
Process 2 has been allocated resources and finished.

System is in a safe state.

Safe Sequence: 1 3 4 0 2
```