WEEK 6

To simulate bankers algorithm for deadlock avoidance

```
#include <stdio.h>

#define MAX_PROCESS 10
#define MAX_RESOURCE 10

int main() {
    int processes, resources;
    int allocation[MAX_PROCESS][MAX_RESOURCE];
    int max_resources[MAX_PROCESS][MAX_RESOURCE];
    int available[MAX_RESOURCE];
    int need[MAX_PROCESS][MAX_RESOURCE];
    int work[MAX_RESOURCE];
    int finish[MAX_PROCESS];

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

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

    printf("Enter the allocation matrix:\n");
    for (int i = 0; i < processes; i++) {
        for (int j = 0; j < resources; j++) {
            scanf("%d", &allocation[i][j]);
        }
    }

    printf("Enter the maximum resources matrix:\n");
    for (int i = 0; i < processes; i++) {
        for (int j = 0; j < resources; j++) {
            scanf("%d", &max_resources[i][j]);
        }
    }

    printf("Enter the available resources:\n");
    for (int j = 0; j < resources; j++) {
        scanf("%d", &available[j]);
    }

    // Calculate the need matrix
    for (int i = 0; i < processes; i++) {
        for (int j = 0; j < resources; j++) {
```

```
        need[i][j] = max_resources[i][j] - allocation[i][j];
    }
}

// Initialize work and finish arrays
for (int j = 0; j < resources; j++) {
    work[j] = available[j];
}

for (int i = 0; i < processes; i++) {
    finish[i] = 0;
}

int count = 0;
int safe_sequence[MAX_PROCESS];

// Banker's Algorithm
while (count < processes) {
    int found = 0;
    for (int i = 0; i < processes; i++) {
        if (finish[i] == 0) {
            int j;
            for (j = 0; j < resources; j++) {
                if (need[i][j] > work[j])
                    break;
            }
            if (j == resources) {
                for (int k = 0; k < resources; k++) {
                    work[k] += allocation[i][k];
                }
                safe_sequence[count] = i;
                finish[i] = 1;
                count++;
                found = 1;
            }
        }
    }
    if (!found) {
        printf("System is in an unsafe state. Deadlock detected.\n");
        return 0;
    }
}

// If the code reaches this point, the system is in a safe state.
```

```
    printf("System is in a safe state. Safe sequence: ");
    for (int i = 0; i < processes; i++) {
        printf("P%d ", safe_sequence[i+1]);
    }
    printf("\n");

    return 0;
}
```

OUTPUT: