LAB PROGRAM 7

Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance.

INPUT

```c
#include <stdio.h>
#include <stdbool.h>
#define MAX_PROCESSES 10
#define MAX_RESOURCES 10

// Function to check if the system is in a safe state
bool isSafeState(int processes, int resources, int available[], int max[][MAX_RESOURCES],
int allocation[][MAX_RESOURCES]){
    int work[MAX_RESOURCES];
    bool finish[MAX_PROCESSES] = {0};
    int safeSequence[MAX_PROCESSES];
    int need[MAX_PROCESSES][MAX_RESOURCES];
    // Calculate the need matrix
    for (int i = 0; i < processes; i++) {
        for (int j = 0; j < resources; j++) {
            need[i][j] = max[i][j] - allocation[i][j];
        }
    }

    for (int i = 0; i < resources; i++) {
        work[i] = available[i];
    }

    int count = 0;

    while (count < processes) {
        bool found = false;
        for (int p = 0; p < processes; p++) {
            if (!finish[p]) {
                bool canProceed = true;
                for (int r = 0; r < resources; r++) {
                    if (need[p][r] > work[r]) {
                        canProceed = false;
                        break;
                    }
                }
                if (canProceed) {
                    printf("P%d is visited ( ", p);
                    for (int r = 0; r < resources; r++) {
                        printf("%d ", work[r]);
                    }
                    printf(")\n");

                    for (int r = 0; r < resources; r++) {
```

```
                    work[r] += allocation[p][r];
                }
                safeSequence[count++] = p;
                finish[p] = true;
                found = true;
            }
        }
    }
    if (!found) {
        printf("System is not in a safe state.\n");
        return false;
    }
}

printf("SYSTEM IS IN SAFE STATE\nThe Safe Sequence is -- (");
for (int i = 0; i < processes; i++) {
    printf("P%d ", safeSequence[i]);
}
printf(")\n");

printf("\nProcess\tAllocation\tMax\t\tNeed\n");
for (int i = 0; i < processes; i++) {
    printf("P%d\t", i);
    for (int j = 0; j < resources; j++) {
        printf("%d ", allocation[i][j]);
    }
    printf("\t\t");
    for (int j = 0; j < resources; j++) {
        printf("%d ", max[i][j]);
    }
    printf("\t\t");
    for (int j = 0; j < resources; j++) {
        printf("%d ", need[i][j]);
    }
    printf("\n");
}

return true;
}

int main() {
    int processes, resources;
    int available[MAX_RESOURCES];
    int max[MAX_PROCESSES][MAX_RESOURCES];
```

```
    int allocation[MAX_PROCESSES][MAX_RESOURCES];

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

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

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

    for (int i = 0; i < processes; i++) {
        printf("Enter details for P%d\n", i);
        printf("Enter allocation -- ");
        for (int j = 0; j < resources; j++) {
            scanf("%d", &allocation[i][j]);
        }
        printf("Enter Max -- ");
        for (int j = 0; j < resources; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    isSafeState(processes, resources, available, max, allocation);

    return 0;
}
```

OUTPUT

```
Enter number of processes: 5
Enter number of resources: 3
Enter Available Resources --
3 3 2
Enter details for P0
Enter allocation -- 0 1 0
Enter Max -- 7 5 3
Enter details for P1
Enter allocation -- 2 0 0
Enter Max -- 3 2 2
Enter details for P2
Enter allocation -- 3 0 2
Enter Max -- 9 0 2
Enter details for P3
Enter allocation -- 2 1 1
Enter Max -- 2 2 2
Enter details for P4
Enter allocation -- 0 0 2
Enter Max -- 4 3 3
P1 is visited ( 3 3 2 )
P3 is visited ( 5 3 2 )
P4 is visited ( 7 4 3 )
P0 is visited ( 7 4 5 )
P2 is visited ( 7 5 5 )
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- (P1 P3 P4 P0 P2 )

Process Allocation      Max             Need
P0      0 1 0           7 5 3           7 4 3
P1      2 0 0           3 2 2           1 2 2
P2      3 0 2           9 0 2           6 0 0
P3      2 1 1           2 2 2           0 1 1
P4      0 0 2           4 3 3           4 3 1


...Program finished with exit code 0
Press ENTER to exit console.
```