NAME : ADITYA RAJ PANDIT

REG NO : 23BRS1157

Develop a banker's algorithm for n processes and m resources. Assume that max, allocation and available matrix are given.

BankersAlgo.c 2

BankersAlgo.c > isSafeState(int, int, int [][10], int [][10], int [], int [])

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

void printMatrix(int matrix[][10], int n, int m) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

void calculateNeedMatrix(int need[][10], int max[][10], int alloc[][10], int n, int m) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            need[i][j] = max[i][j] - alloc[i][j];
        }
    }
}

bool isSafeState(int n, int m, int alloc[][10], int need[][10], int avail[], int safeSeq[]) {
    int work[10], finish[10] = {0};
    for (int i = 0; i < m; i++) work[i] = avail[i];

    int count = 0;
    while (count < n) {
        bool found = false;
        for (int p = 0; p < n; p++) {
            if (!finish[p]) {
                int j;
                for (j = 0; j < m; j++) {
                    if (need[p][j] > work[j])
                        break;
                }
                if (j == m) {
                    for (int k = 0; k < m; k++)
                        work[k] += alloc[p][k];
```

BankersAlgo.c 2

BankersAlgo.c > isSafeState(int, int, int [][10], int [][10], int [], int [])

```c
bool isSafeState(int n, int m, int alloc[][10], int need[][10], int avail[], int safeSeq[]) {
                    if (need[p][j] > work[j])
                        break;
                }
                if (j == m) {
                    for (int k = 0; k < m; k++)
                        work[k] += alloc[p][k];
                    safeSeq[count++] = p;
                    finish[p] = 1;
                    found = true;
                }
            }
        }
        if (!found) return false; // No safe sequence
    }
    return true;
}

bool canGrantRequest(int p, int m, int request[], int alloc[][10], int need[][10], int avail[]) {
    for (int i = 0; i < m; i++) {
        if (request[i] > need[p][i] || request[i] > avail[i])
            return false;
    }
    return true;
}

void grantRequest(int p, int m, int request[], int alloc[][10], int need[][10], int avail[]) {
    for (int i = 0; i < m; i++) {
        avail[i] -= request[i];
        alloc[p][i] += request[i];
        need[p][i] -= request[i];
    }
}

int main() {
    int n, m;
    printf("Enter number of processes: ");
```

BankersAlgo.c 2

BankersAlgo.c > isSafeState(int, int, int [][10], int [][10], int [], int [])

```c
}

int main() {
    int n, m;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter number of resources: ");
    scanf("%d", &m);

    int alloc[10][10], max[10][10], avail[10];
    printf("Enter allocation matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &alloc[i][j]);
        }
    }

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

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

    int need[10][10];
    calculateNeedMatrix(need, max, alloc, n, m);
    printf("Need matrix is:\n");
    printMatrix(need, n, m);

    int safeSeq[10];
    if (isSafeState(n, m, alloc, need, avail, safeSeq)) {
        printf("System is in a safe state. Safe sequence: ");
```

```c
C BankersAlgo.c > isSafeState(int, int, int [][10], int [][10], int [], int [])
int main() {
    printMatrix(need, n, m);

    int safeSeq[10];
    if (isSafeState(n, m, alloc, need, avail, safeSeq)) {
        printf("System is in a safe state. Safe sequence: ");
        for (int i = 0; i < n; i++)
            printf("P%d ", safeSeq[i]);
        printf("\n");
    } else {
        printf("System is not in a safe state.\n");
    }

    int request1[10];
    printf("Enter request for process P1: ");
    for (int i = 0; i < m; i++) {
        scanf("%d", &request1[i]);
    }

    if (canGrantRequest(1, m, request1, alloc, need, avail)) {
        grantRequest(1, m, request1, alloc, need, avail);
        if (isSafeState(n, m, alloc, need, avail, safeSeq)) {
            printf("Request can be granted. System is in a safe state. Final available resources:\n");
            for (int i = 0; i < m; i++) printf("%d ", avail[i]);
            printf("\n");
        } else {
            printf("Request would lead to an unsafe state.\n");
        }
    } else {
        printf("Request cannot be granted immediately.\n");
    }

    int request2[10];
    printf("Enter request for process P2: ");
    for (int i = 0; i < m; i++) {
        scanf("%d", &request2[i]);
    }
```

```c
    }

    int request2[10];
    printf("Enter request for process P2: ");
    for (int i = 0; i < m; i++) {
        scanf("%d", &request2[i]);
    }

    if (canGrantRequest(2, m, request2, alloc, need, avail)) {
        grantRequest(2, m, request2, alloc, need, avail);
        if (isSafeState(n, m, alloc, need, avail, safeSeq)) {
            printf("Request can be granted. System is in a safe state. Final available resources:\n");
            for (int i = 0; i < m; i++) printf("%d ", avail[i]);
            printf("\n");
        } else {
            printf("Request would lead to an unsafe state.\n");
        }
    } else {
        printf("Request cannot be granted immediately.\n");
    }

    return 0;
}
```

For the set of inputs :

No of processes : 5

No of resources : 4

Allocation matrix : 0 1 0 3

2 0 0 1

3 0 2 1

2 1 1 0

0 0 2 0

Maximum matrix : 7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Available resouces : 3 3 2 1

1) Find the need matrix

```
c:\Codes\OS>cd "c:\Codes\OS\" && gcc BankersAlgo.c -o BankersAlgo && "c:\Codes\OS\"BankersAlgo
Enter number of processes: 5
Enter number of resources: 4
Enter allocation matrix:
0 1 0 3
2 0 0 1
3 0 2 1
2 1 1 0
0 0 2 0
Enter maximum matrix:
7 5 3 4
3 2 2 2
9 0 2 2
2 2 2 2
4 3 3 1
Enter available resources:
3 3 2 1
Need matrix is:
7 4 3 1
1 2 2 1
6 0 0 1
0 1 1 2
4 3 1 1
System is in a safe state. Safe sequence: P1 P3 P4 P0 P2
```

2) Check the system is in a safe state

```
Need matrix is:
7 4 3 1
1 2 2 1
6 0 0 1
0 1 1 2
4 3 1 1
System is in a safe state. Safe sequence: P1 P3 P4 P0 P2
```

3) if a request from p1 arrives for (A1,B1,C1,D1), can the requested be granted immediately? will it lead safe state? if it is safe state print the final available matrix .

```
Enter request for process P1: 1 0 2 0
Request can be granted. System is in a safe state. Final available resources:
2 3 0 1
```

4) if a request from p2 arrives for (A2,B2,C2,D2) can the requested be granted immediately? will it lead safe state? if it is safe state print the final available matrix.

```
Enter request for process P1: 1 0 2 0
Request can be granted. System is in a safe state. Final available resources:
2 3 0 1
Enter request for process P2: 0 2 0 0
Request cannot be granted immediately.
```

In this case request from P2 is 0 2 0 0 and the available resources are 2 3 0 1. Even though the availability fulfils the need of P2 it would not lead to a safe state. Therefore request cannot be granted.

For the set of inputs :

No of processes : 5

No of resources : 3

Allocation Matrix : 0 1 0

2 0 0

3 0 2

2 1 1

0 0 2

Maximum Matrix : 7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Available Resources : 1 1 2

1) Need Matrix :

```
Enter number of processes: 5
Enter number of resources: 3
Enter allocation matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter maximum matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter available resources:
1 1 2
Need matrix is:
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1
System is not in a safe state.
```

2) Check the system is in a safe state

```
System is not in a safe state.
```

3) **if a request from p1 arrives for (A1,B1,C1,D1), can the requested be granted immediately? will it lead safe state? if it is safe state print the final available matrix .**

```
System is not in a safe state.
Enter request for process P1: 0 2 0
Request cannot be granted immediately.
Enter request for process P2: 1 2 3
Request cannot be granted immediately.
```