

OPERATING SYSTEM - CS23431

EX 9

DEADLOCK AVOIDANCE

NAME: MOUNAMITHRA ROLL NO:230701197

PROGRAM:

```
#include <stdio.h>

int main() {
    int resource, process;

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

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

    int inst[resource];

    printf("Enter max instance of each resource: ");
    for (int i = 0; i < resource; i++) {
        scanf("%d", &inst[i]);
    }

    int allocated[process][resource], max[process][resource], need[process][resource];

    int available[resource];

    printf("Enter allocated matrix row-wise:\n");

    for (int i = 0; i < process; i++) {
        printf("Process %d: ", i + 1);
        for (int j = 0; j < resource; j++) {
            scanf("%d", &allocated[i][j]);
        }
    }
}
```

```
}
```

```
printf("Enter Max matrix row-wise:\n");
```

```
for (int i = 0; i < process; i++) {
```

```
    printf("Process %d: ", i + 1);
```

```
    for (int j = 0; j < resource; j++) {
```

```
        scanf("%d", &max[i][j]);
```

```
    }
```

```
}
```

```
for (int i = 0; i < process; i++) {
```

```
    for (int j = 0; j < resource; j++) {
```

```
        need[i][j] = max[i][j] - allocated[i][j];
```

```
    }
```

```
}
```

```
for (int j = 0; j < resource; j++) {
```

```
    int sum = 0;
```

```
    for (int i = 0; i < process; i++) {
```

```
        sum += allocated[i][j];
```

```
    }
```

```
    available[j] = inst[j] - sum;
```

```
}
```

```
int finish[process];
```

```
for (int i = 0; i < process; i++) {
```

```
    finish[i] = 0;
```

```
}
```

```
int safeseq[process];
```

```
int count = 0, canrun, notsafe = 0;
```

```
while (count < process) {
```

```
    int found = 0;
```

```
    for (int i = 0; i < process; i++) {
```

```
        if (!finish[i]) {
```

```
            canrun = 1;
```

```
            for (int j = 0; j < resource; j++) {
```

```
                if (need[i][j] > available[j]) {
```

```
                    canrun = 0;
```

```
                    break;
```

```
                }
```

```
            }
```

```
            if (canrun) {
```

```
                for (int j = 0; j < resource; j++) {
```

```
                    available[j] += allocated[i][j];
```

```
                }
```

```
                safe_seq[count++] = i;
```

```
                finish[i] = 1;
```

```
                found = 1;
```

```
            }
```

```
        }
```

```
    }
```

```
    if (!found) {
```

```
        printf("System is not in safe sequence\n");
```

```
        notsafe = 1;
```

```
        break;
```

```
    }
```

```
}
```

```
if (!notsafe) {  
    printf("The system is in a safe sequence:\n");  
    for (int i = 0; i < process; i++) {  
        printf("P%d", safeseq[i]);  
        if (i != process - 1) {  
            printf(" -> ");  
        }  
    }  
    p r i n t f ( " \ n " ) ;  
}  
  
return 0;  
}
```

OUTPUT:

```
Enter number of resources: 3
Enter number of processes: 5
Enter max instance of each resource: 10
5
7
Enter allocated matrix row-wise:
Process 1: 0
1
0
Process 2: 2
0
0
Process 3: 3
0
2
Process 4: 2
1
1
Process 5: 0
0
2
Enter Max matrix row-wise:
Process 1: 7
5
3
Process 2: 3
2
2
Process 3: 9
0
2
Process 4: 4
2
2
Process 5: 5
3
3
The system is in a safe sequence:
P1 -> P3 -> P4 -> P0 -> P2
```