

Write a C program to simulate deadlock detection.

```
# include <cs101.h>
# define MAX-P 10
# define MAX-R 10
int num-process, num-resources;
int available[MAX-R];
int max[MAX-P][MAX-R];
int allocation[MAX-P][MAX-R];
int need[MAX-P][MAX-R];
void main()
{
    int process-order[MAX-P];
    inputData();
    int flag = checkSafety(process-order);
    if (flag == 1)
        printf("Deadlock is not present");
    else
        printf("Deadlock is detected \n");
}

int checkSafety(int process-order[])
{
    int work[MAX-R];
    int finish[MAX-P] = {0};
    for (int i = 0; i < num-resources; i++)
        work[i] = available[i];
    int completed = 0;
    while (completed < num-processes)
    {
        int found = 0;
```

Date \_\_\_\_\_  
Page \_\_\_\_\_

```
for (int i = 0; i < num_processes; i++)
```

```
    if (!finish(i))
```

```
        int j;
```

```
        for (j = 0; j < num_resources; j++)
```

```
            if (need[i][j] > work[j])
```

```
                break;
```

```
}
```

```
if (j == num_resources)
```

```
    for (int k = 0; k < num_resources; k++)
```

```
        work[k] += allocation[i][k];
```

```
}
```

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

```
process_order[completed] = i,
```

```
completed++;
```

```
found = 1;
```

```
j
```

```
if (!found)
```

```
    exit(0);
```

```
y
```

```
void inputData()
```

```
{
```

```
    printf("Enter the number of processes: ");
```

```
    scanf("%d", &num_processes);
```

```
    printf("Enter the number of resources: ");
```

```
    scanf("%d", &num_resources);
```

```
    printf("Enter the available resources: ");
```

```
    for (int i = 0; i < num_resources; i++)
```

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

```
}
```

```
printf ("Enter the request matrix : \n");
for (int i = 0; i < num_processes; i++)
    for (int j = 0; j < num_resources; j++)
        scanf ("%d", &max[i][j]);
```

```
printf ("Enter the allocation matrix : \n");
for (int i = 0; i < num_processes; i++)
    for (int j = 0; j < num_resources; j++)
        scanf ("%d", &allocation[i][j]);
    need[i][j] = max[i][j] - allocation[i][j];
```

Output :

Enter the number of processes: 5

Enter the number of resources: 3

Enter the available resources:

3 3 2

Enter the request matrix:

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Enter the allocation matrix:

0 1 0

1 0 0

3 0 3

2 1 1

0 0 2

Date \_\_\_\_\_

Page \_\_\_\_\_

Deadlock is not present

Condition of deadlock is not met

(1) Mutual Exclusion Condition

Process P1 holds R1 & P2 holds R2

P1 can't access R2 because P2 holds R2

Condition of deadlock is not met

(2) Hold and Wait Condition

P1 holds R1 & wants R2

P2 holds R2 & wants R1

Condition of deadlock is not met

(3) No Preemption Condition

Condition of deadlock is not met

(4) Circular Wait Condition

P1 -> P2 -> P3 -> P4 -> P1

Condition of deadlock is not met

(5) Resource Request Not Granted Condition

P1 -> P2 -> P3 -> P4 -> P1

Condition of deadlock is not met

(6) Resource Acquisition Order Condition

P1 -> P2 -> P3 -> P4 -> P1

Condition of deadlock is not met

(7) Resource Availability Condition

P1 -> P2 -> P3 -> P4 -> P1

Condition of deadlock is not met

(8) Resource Maximum Limit Condition

P1 -> P2 -> P3 -> P4 -> P1

Condition of deadlock is not met

(9) Resource Minimum Limit Condition

P1 -> P2 -> P3 -> P4 -> P1

Condition of deadlock is not met

(10) Resource Maximum Limit Condition

P1 -> P2 -> P3 -> P4 -> P1

Condition of deadlock is not met

(11) Resource Minimum Limit Condition

P1 -> P2 -> P3 -> P4 -> P1

Condition of deadlock is not met

```
C:\Users\Avani\Desktop\1bm21cs036\dd\bin\Debug\dd.exe
S
Enter the number of processes: 5
Enter the number of resources: 3
Total Amount of Resource R 1: 3
Total Amount of Resource R 2: 3
Total Amount of Resource R 3: 2

Enter the request matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3

Enter the allocation matrix:
0 1 0
2 0 0
3 0 3
2 1 1
0 0 2

Deadlock detected
```