```c
//Bankers Algorithm to check for safe state.
#include <stdio.h>

struct Process
{
    int id;
    int max_need;
    int allocated;
    int remaining;
    int flag;
};

int available;
int total;
int count=0;
int CHECK(Process P[], int n)
{
    for(int i=0 ; i<n ; i++)
    {
        if(P[i].flag == 0)
            return 0;
    }
    return 1;
}

int isSafe(Process P[], int n)
{
    count++;
    if (CHECK(P,n))
    {
        return 1;
    }
    if(count > n){
        printf("DEADLOCK\n");
        printf("NOT SAFE!\n");
        return 0;
    }
    for(int i=0 ; i<n ; i++)
    {
        if(P[i].flag==0 && P[i].remaining<available)
        {
            P[i].flag = 1;
            printf("P%d\t", P[i].id);
            available+=P[i].allocated;
        }
    }
    return isSafe(P,n);
}

int main()
{
    int n,sum=0;
    printf("Enter number processes : ");
    scanf("%d", &n);
    Process P[n];
    printf("Enter data for each Process : \n");
    for(int i=0 ; i<n ; i++)
    {
        printf("ID : ");
        scanf("%d", &P[i].id);
        printf("Max Needed : ");
        scanf("%d", &P[i].max_need);
        printf("Allocated : ");
        scanf("%d", &P[i].allocated);
        sum+=P[i].allocated;
        P[i].remaining = P[i].max_need - P[i].allocated;
        P[i].flag = 0;
        printf("\n");
    }
```

```
    printf("Total  Instances : ");
    scanf("%d", &total);
    available = total - sum;
    isSafe(P,n);
}
/*
OUTPUT

Enter number processes : 5
Enter data for each Process :
ID : 1
Max Needed : 5
Allocated : 2

ID : 2
Max Needed : 6
Allocated : 3

ID : 3
Max Needed : 6
Allocated : 2

ID : 4
Max Needed : 2
Allocated : 1

ID : 5
Max Needed : 4
Allocated : 1

Total  Instances : 10
DEADLOCK
NOT SAFE!
*/
```