Write a C-program to simulate deadlock detection.

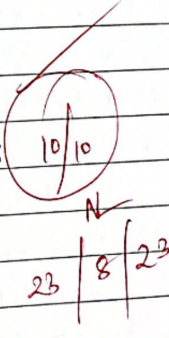
```
#include <stdio.h>
static int mark [20];
int i, j, np, nr;
int main()
{
int alloc [10][10], request [20][10], avail [10], r[10], w[10];
Printf("\n Enter the no. of process: ");
scanf("%d", &np);
Printf("\n Enter the no. of resources");
scanf("%d", &nr);
for(i=0; i<nr; i++)
{
Printf("\n Total Amnt of resource R %d:", i+1);
scanf("%d", &r[i]);
}
Printf("Enter the request matrix:");
for(i=0; i<np; i++)
for(j=0; j<nr; j++)
scanf("%d", &request[i][j]);
Printf("\n Enter the allocation matrix:");
for(i=0; i<np; i++)
for(j=0; j<nr; j++)
scanf("%d", &alloc[i][j]);
for(i=0; j<nr; j++)
Scanf("%d", &alloc[i][j]);
for(j=0; j<nr; j++)
{
avail[j]=r[j];
for(i=0; i<np; i++)
{
avail[j]=alloc[i][j];
}}
```

```
for (i=0; i<np; i++)
 {
 int count=0;
 for (j=0; j<nr; j++)
 {
 if (alloc[i][j]==0)
  count++;
 else
 break;
 }
 if (count==nr)
 mark[i]=1;
 }
 for (j=0; j<nr; j++)
 w[j]=avail[j];
 for (i=0; i<np; i++)
 {
 int canbeprocessed=0;
 if (mark[i]!=1)
 {
 for (j=0; j<nr; j++)
 {
 if (request[i][j]<=w[j])
  can be processed=1;
 else
 {
 can be processed=0;
 break;
 }
 }
 if (can be processed)
 {
 mark[i]=1;
 for (j=0; j<nr; j++)
```

```
w[j]+=alloc[i][j];
}
}
}
int deadlock=0;
for(i=0;i<np;i++)
 if(mark[i]!=1)
 deadlock=1;
if(deadlock)
Printf("\n Deadlock detected");
else
Printf("\n No deadlock possible");
}
```

Output

Enter the no. of process
3
Enter the no. of resource
3
Enter the request matrix
7 5 3
3 2 2
9 0 2
Enter the allocation matrix
0 1 0
2 0 0
3 0 2
Enter available resources
3 3 2
No ~~detec~~ deadlock possible.



```
PS D:\VS Code\OS> cd "d:\VS Code\OS\" ; if ($?) { gcc bankersV2.c -o bankersV2 } ; if ($?) { .\bankersV2 }
Enter number of processes and number of resources required
3 3
Enter total number of required resources 3 for each process
7 5 3
3 2 2
9 0 2
Enter number of allocated resources 3 for each process
0 1 0
2 0 0
3 0 2
Enter number of available resources
1 2 3
```