Lab 7:
Deadlock in OS

Programme	:	**BTech. CSE Core**	Semester	:	**Win 2021-22**
Course	:	**Operating Systems**	Code	:	**CSE2005**
Faculty	:	**Dr. Shyamala L**	Slot	:	**L25+L26**
Name	:	**Hariket Sukesh Kumar Sheth**	Register No.	:	**20BCE1975**

LAB 7

Sample Question:

1. Consider the following snapshot of a system in which four resources A, B, C and D are available. The system currently contains 6 instances of A, 4 of resource B, 4 of resource C, 2 resources D after allocation.

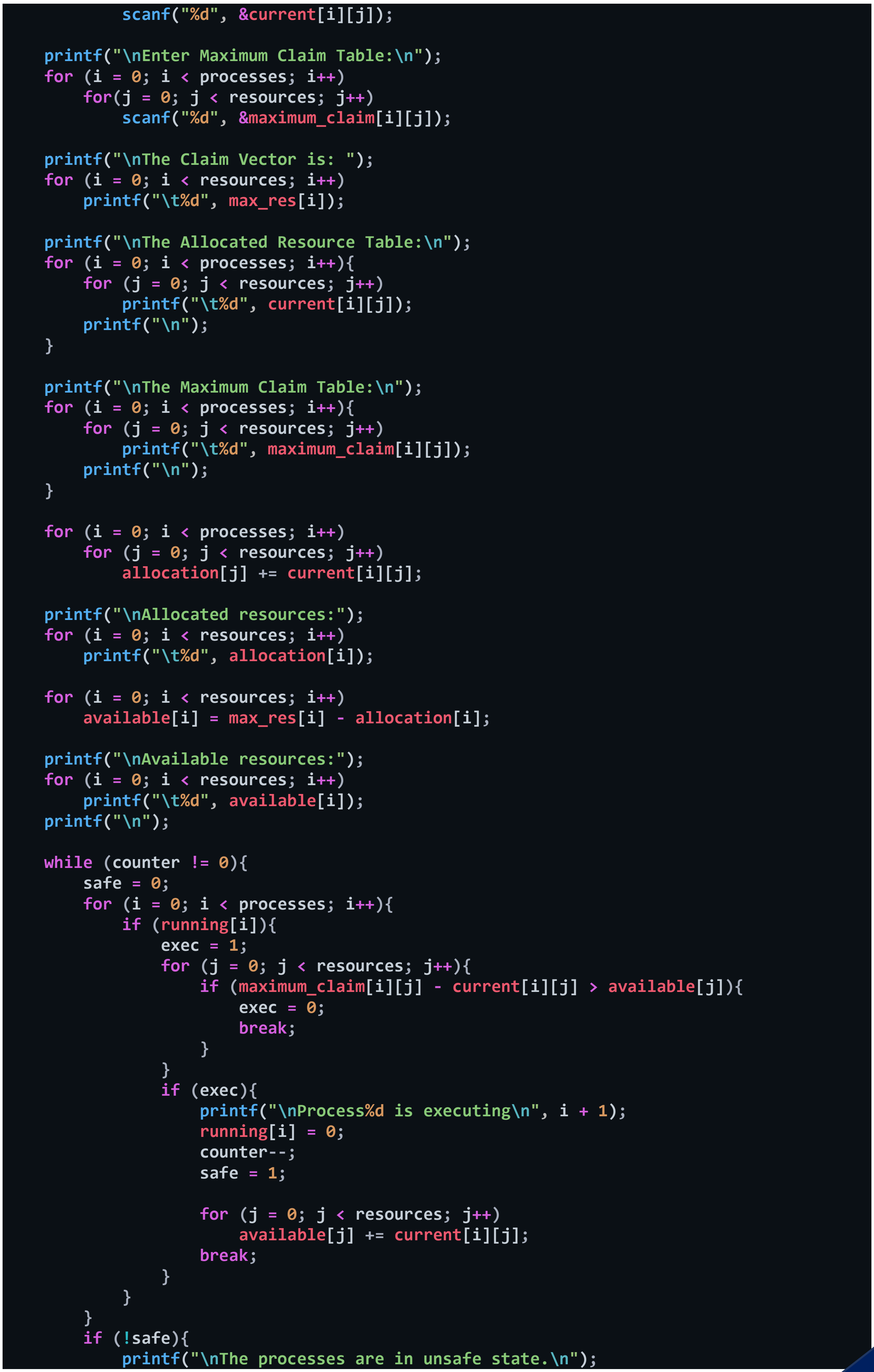
Write a C/ C++ code to do deadlock avoidance using banker's algorithm:

- Compute what each process might still request and fill this in under the column Need.
- Is the system in a safe state? Why or why not?
- Is the system deadlocked? Why or why not?

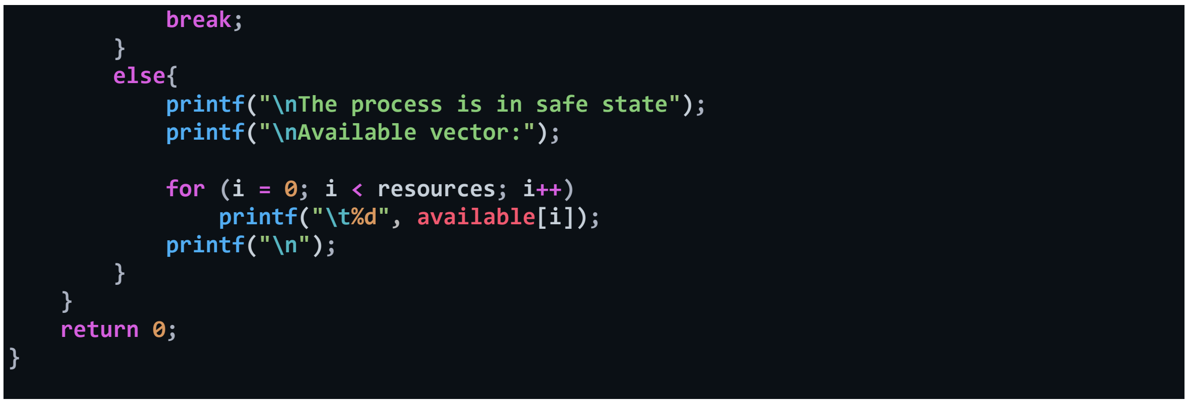
	Allocation				*Max*				*Need*				*Available*			
	A	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*
P_0	2	0	1	1	3	2	1	1					6	4	4	2
P_1	1	1	0	0	1	2	0	2								
P_2	1	0	1	0	3	2	1	0								
P_3	0	1	0	1	2	1	0	1								

CODE:

```c
#include <stdio.h>

int current[5][5], maximum_claim[5][5], available[5];
int allocation[5] = {0, 0, 0, 0, 0};
int max_res[5], running[5], safe = 0;
int counter = 0, i, j, exec, resources, processes, k = 1;

int main(){
    printf("\nEnter number of processes: ");
    scanf("%d", &processes);

    for (i = 0; i < processes; i++0){
        running[i] = 1;
        counter++;
    }

    printf("\nEnter number of resources: ");
    scanf("%d", &resources);

    printf("\nEnter Claim Vector:");
    for (i = 0; i < resources; i++)
        scanf("%d", &max_res[i]);

    printf("\nEnter Allocated Resource Table:\n");
    for (i = 0; i < processes; i++)
        for(j = 0; j < resources; j++)
```

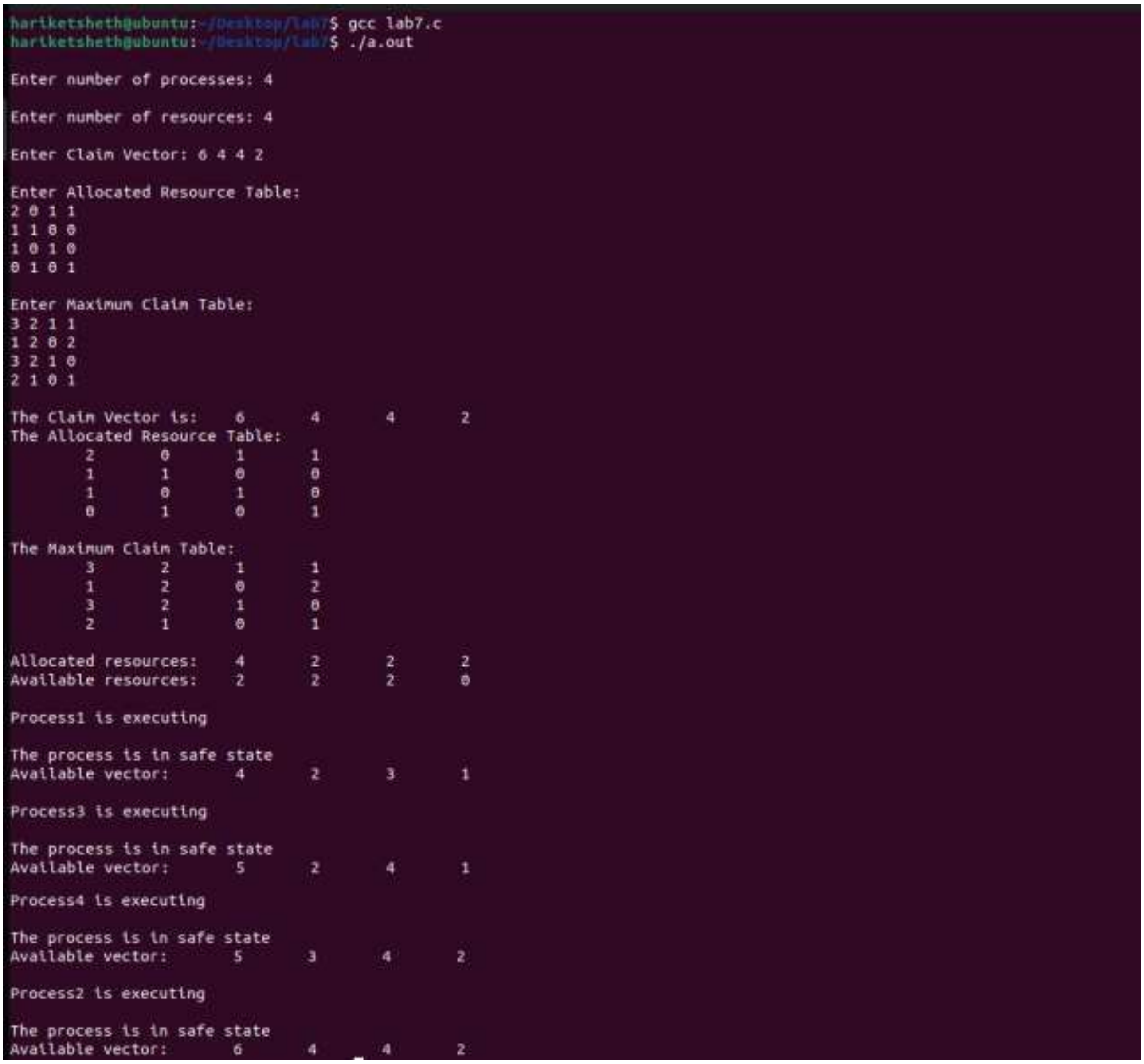


```
            scanf("%d", &current[i][j]);

    printf("\nEnter Maximum Claim Table:\n");
    for (i = 0; i < processes; i++)
        for(j = 0; j < resources; j++)
            scanf("%d", &maximum_claim[i][j]);

    printf("\nThe Claim Vector is: ");
    for (i = 0; i < resources; i++)
        printf("\t%d", max_res[i]);

    printf("\nThe Allocated Resource Table:\n");
    for (i = 0; i < processes; i++){
        for (j = 0; j < resources; j++)
            printf("\t%d", current[i][j]);
        printf("\n");
    }

    printf("\nThe Maximum Claim Table:\n");
    for (i = 0; i < processes; i++){
        for (j = 0; j < resources; j++)
            printf("\t%d", maximum_claim[i][j]);
        printf("\n");
    }

    for (i = 0; i < processes; i++)
        for (j = 0; j < resources; j++)
            allocation[j] += current[i][j];

    printf("\nAllocated resources:");
    for (i = 0; i < resources; i++)
        printf("\t%d", allocation[i]);

    for (i = 0; i < resources; i++)
        available[i] = max_res[i] - allocation[i];

    printf("\nAvailable resources:");
    for (i = 0; i < resources; i++)
        printf("\t%d", available[i]);
    printf("\n");

    while (counter != 0){
        safe = 0;
        for (i = 0; i < processes; i++){
            if (running[i]){
                exec = 1;
                for (j = 0; j < resources; j++){
                    if (maximum_claim[i][j] - current[i][j] > available[j]){
                        exec = 0;
                        break;
                    }
                }
                if (exec){
                    printf("\nProcess%d is executing\n", i + 1);
                    running[i] = 0;
                    counter--;
                    safe = 1;

                    for (j = 0; j < resources; j++)
                        available[j] += current[i][j];
                    break;
                }
            }
        }
        if (!safe){
            printf("\nThe processes are in unsafe state.\n");
```



```
                break;
            }
            else{
                printf("\nThe process is in safe state");
                printf("\nAvailable vector:");

                for (i = 0; i < resources; i++)
                    printf("\t%d", available[i]);
                printf("\n");
            }
        }
    return 0;
}
```

OUTPUT:



```
hariketsheth@ubuntu:~/Desktop/lab7$ gcc lab7.c
hariketsheth@ubuntu:~/Desktop/lab7$ ./a.out

Enter number of processes: 4

Enter number of resources: 4

Enter Claim Vector: 6 4 4 2

Enter Allocated Resource Table:
2 0 1 1
1 1 0 0
1 0 1 0
0 1 0 1

Enter Maximum Claim Table:
3 2 1 1
1 2 0 2
3 2 1 0
2 1 0 1

The Claim Vector is:    6       4       4       2
The Allocated Resource Table:
        2       0       1       1
        1       1       0       0
        1       0       1       0
        0       1       0       1

The Maximum Claim Table:
        3       2       1       1
        1       2       0       2
        3       2       1       0
        2       1       0       1

Allocated resources:    4       2       2       2
Available resources:    2       2       2       0

Process1 is executing

The process is in safe state
Available vector:       4       2       3       1

Process3 is executing

The process is in safe state
Available vector:       5       2       4       1

Process4 is executing

The process is in safe state
Available vector:       5       3       4       2

Process2 is executing

The process is in safe state
Available vector:       6       4       4       2
```