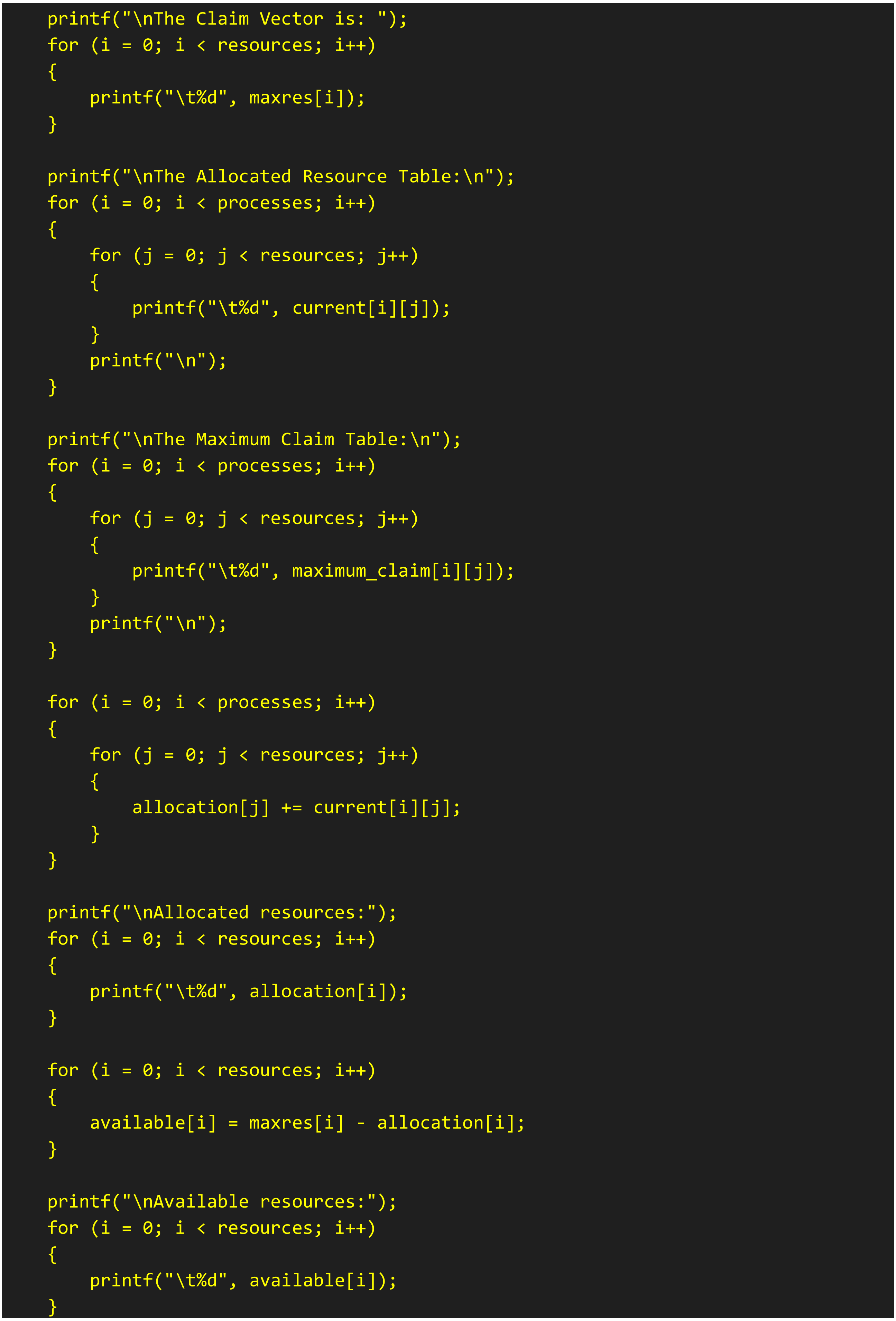
LAB-07 HUZAIFA SALMAN DT-34

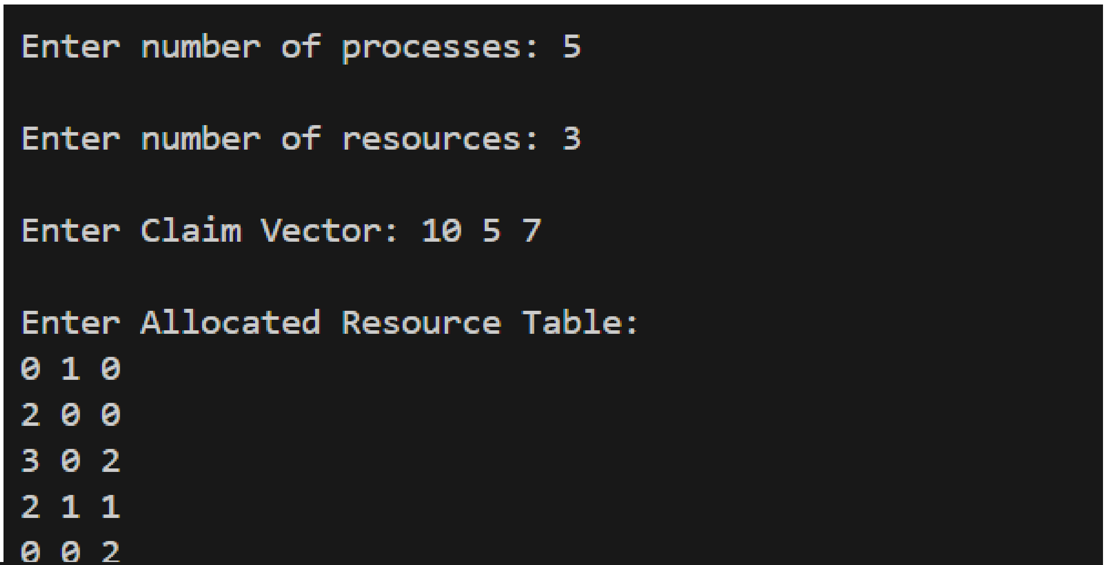
PROGRAM:

```
#include <stdio.h>

int current[5][5], maximum_claim[5][5], available[5];
int allocation[5] = {0, 0, 0, 0, 0};
int maxres[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++)
    {
        running[i] = 1;
        counter++;
    }

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

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

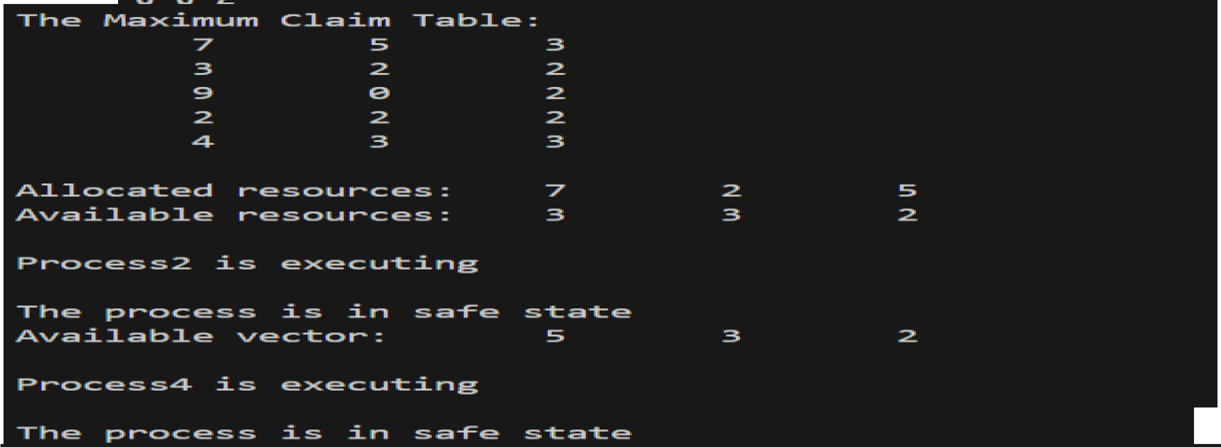


```
    printf("\nThe Claim Vector is: ");
    for (i = 0; i < resources; i++)
    {
        printf("\t%d", maxres[i]);
    }

    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] = maxres[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;
}
```

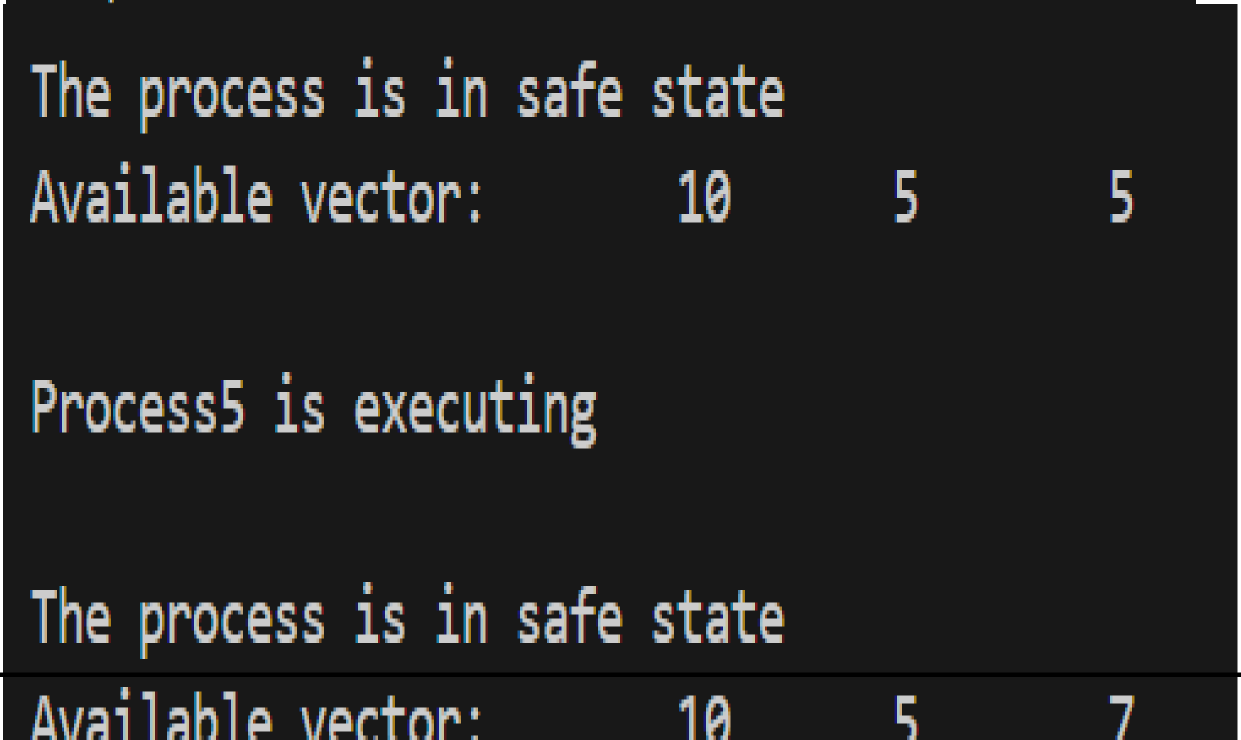
OUPUT SSHOTS :



```
Enter number of processes: 5

Enter number of resources: 3

Enter Claim Vector: 10 5 7

Enter Allocated Resource Table:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
```



```
The Maximum Claim Table:
        7       5       3
        3       2       2
        9       0       2
        2       2       2
        4       3       3

Allocated resources:    7       2       5
Available resources:    3       3       2

Process2 is executing

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

Process4 is executing

The process is in safe state
```



```
The process is in safe state
Available vector:       10      5       5


Process5 is executing


The process is in safe state
Available vector:       10      5       7
```