LAB 07

Course: CT-353-Operating Systems

Department: BCIT (Specialisation in Data Science)

Instructor's Name: Muhammed Abdullah Siddiqui

Student Name: Maryam Ashraff (DT-22050)

CODE:

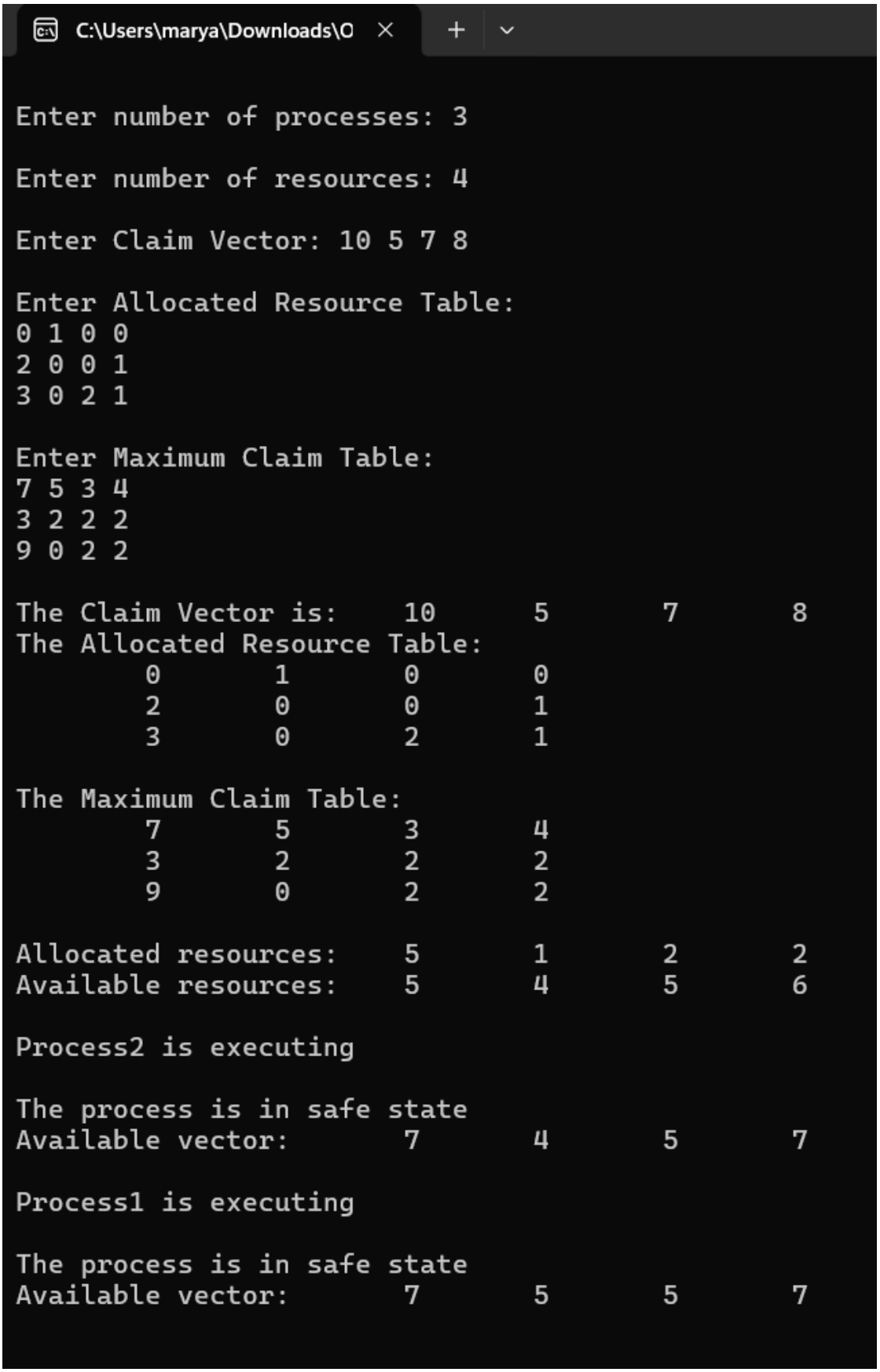
```c
#include <stdio.h>

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];
}
```

```c
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:


C:\Users\marya\Downloads\O
Enter number of processes: 3
Enter number of resources: 4
Enter Claim Vector: 10 5 7 8
Enter Allocated Resource Table:
0 1 0 0
2 0 0 1
3 0 2 1
Enter Maximum Claim Table:
7 5 3 4
3 2 2 2
9 0 2 2
The Claim Vector is: 10 5 7 8
The Allocated Resource Table:
0 1 0 0
2 0 0 1
3 0 2 1
The Maximum Claim Table:
7 5 3 4
3 2 2 2
9 0 2 2
Allocated resources: 5 1 2 2
Available resources: 5 4 5 6
Process2 is executing
The process is in safe state
Available vector: 7 4 5 7
Process1 is executing
The process is in safe state
Available vector: 7 5 5 7

```
Process3 is executing

The process is in safe state
Available vector:       10      5       7       8

--------------------------------
Process exited after 47.61 seconds with return value 0
Press any key to continue . . .
```