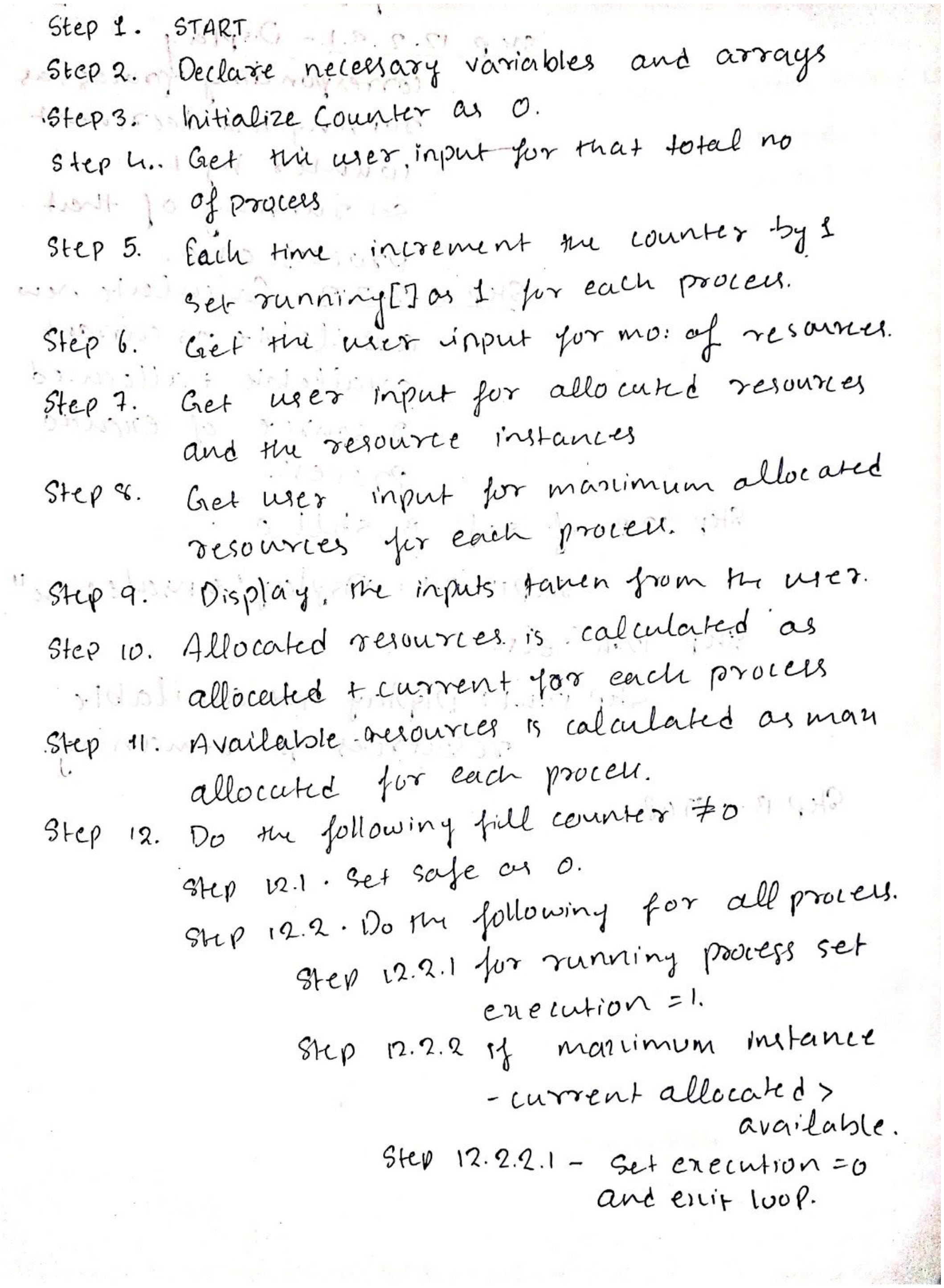
Experiment 3 **Surya Narayanan**

Banker's Algorithm

s5 csb 48

---

1. **Algorithm**

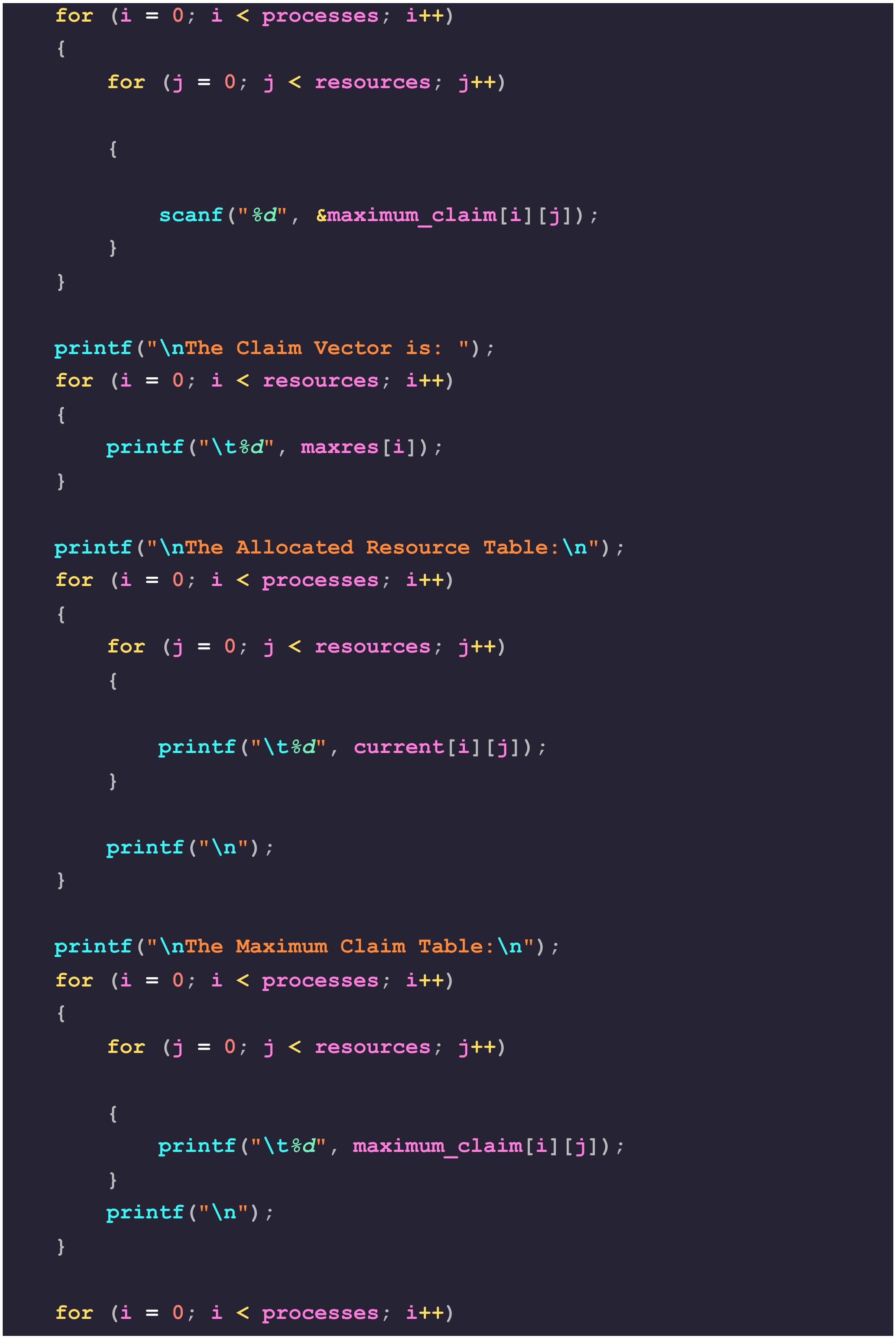


```
Step 1.  START
Step 2.  Declare necessary variables and arrays
Step 3.  Initialize Counter as 0.
Step 4.. Get the user input for that total no
         of process
Step 5.  Each time increment the counter by 1
         set running[] as 1 for each process.
Step 6.  Get the user input for no: of resources.
Step 7.  Get user input for allocated resources
         and the resource instances
Step 8.  Get user input for maximum allocated
         resources for each process.
Step 9.  Display the inputs taken from the user.
Step 10. Allocated resources is calculated as
         allocated + current for each process
Step 11. Available resources is calculated as max
         allocated for each process.
Step 12. Do the following till counter ≠0
         Step 12.1 . Set safe as 0.
         Step 12.2 . Do the following for all process.
                   Step 12.2.1 for running process set
                               execution =1.
                   Step 12.2.2 if maximum instance
                               - current allocated >
                                           available.
                       Step 12.2.2.1 - Set execution =0
                                       and exit loop.
```

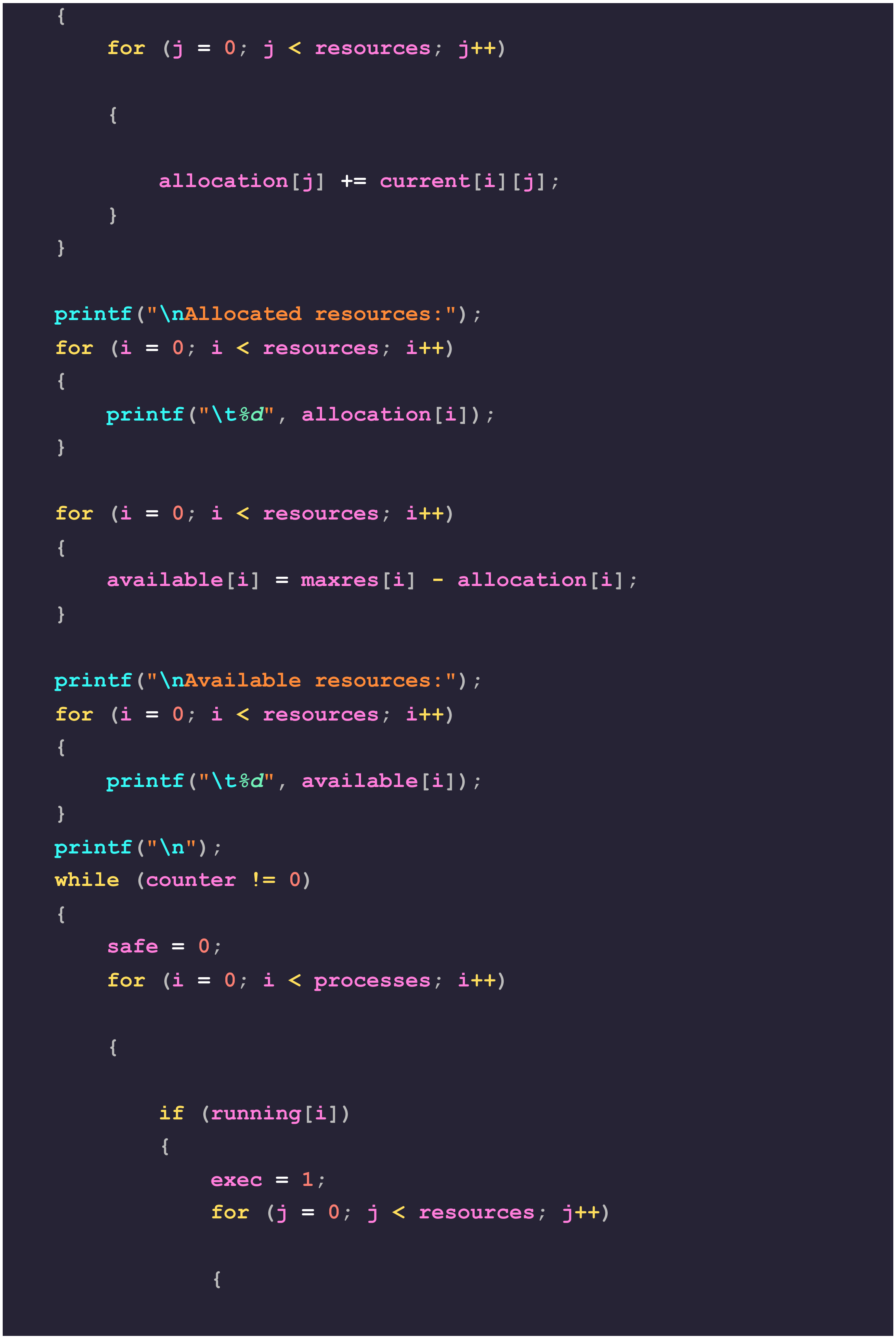
```
            Step 12.2.3- if execution is still 1
                Step 12.2.3.1 - Display corresponding process as
                                running and decrement counter by 1.
                                set running of that process as 0.
                Step 12.2.3.2. Calculate new available as current
                               available + allocated resource of executed
                               process.
    Step 12.3- if safe is still 0
            Step 12.3.1 : Display "unsafe state".
    Step 12.4 : else
            Step 12.4.1 : Display the available
                          resources for remaining.

Step 13.. STOP.
```

Program

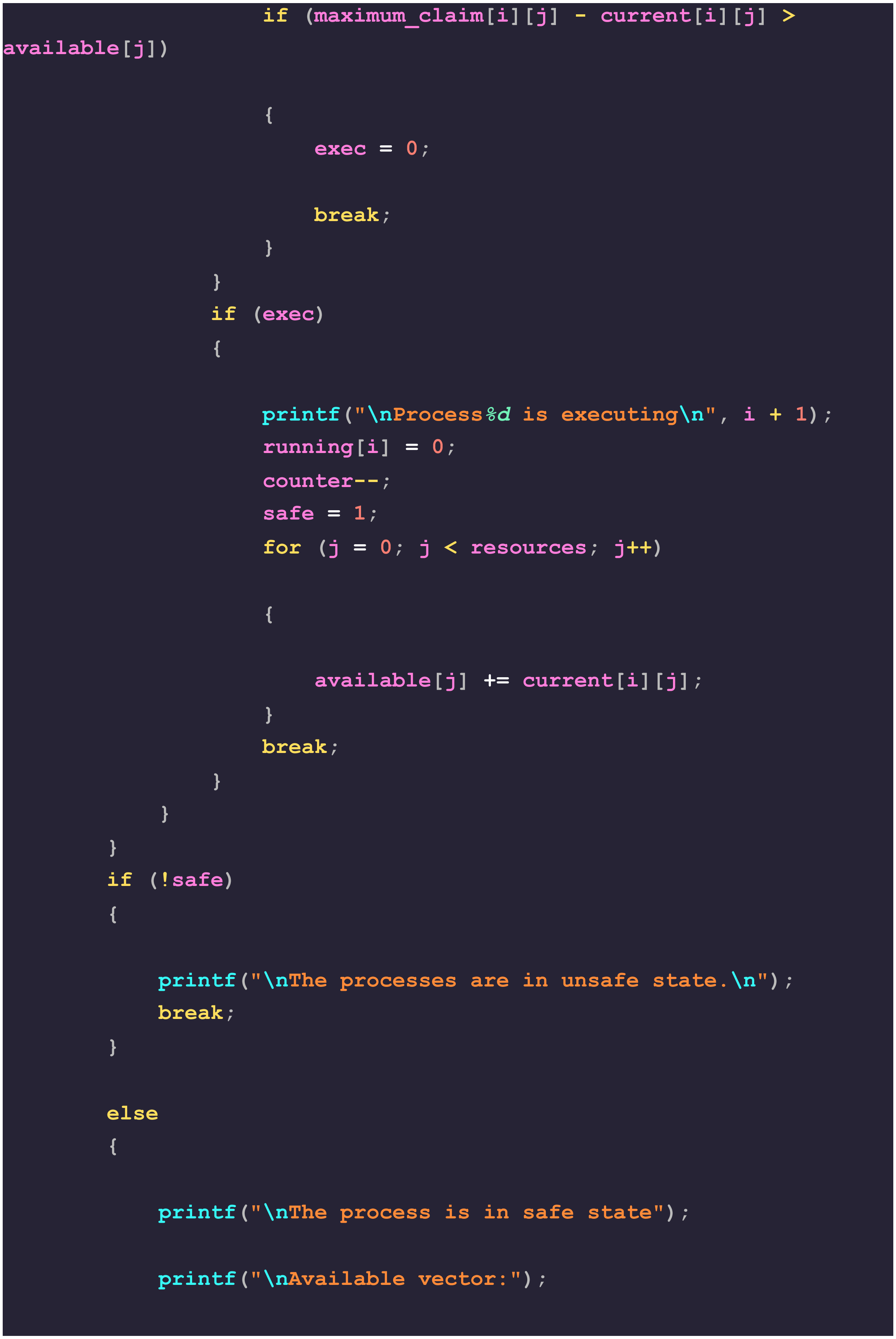
```c
#include <stdio.h>
int current[5][5], maximum_claim[5][5], available[5];
int allocation[5] = {0, 0, 0, 0, 0};
int maxres[5], running[5], safe = 0;
int counter = 0, i, j, exec, resources, processes, k = 1;

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);

    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++)
```



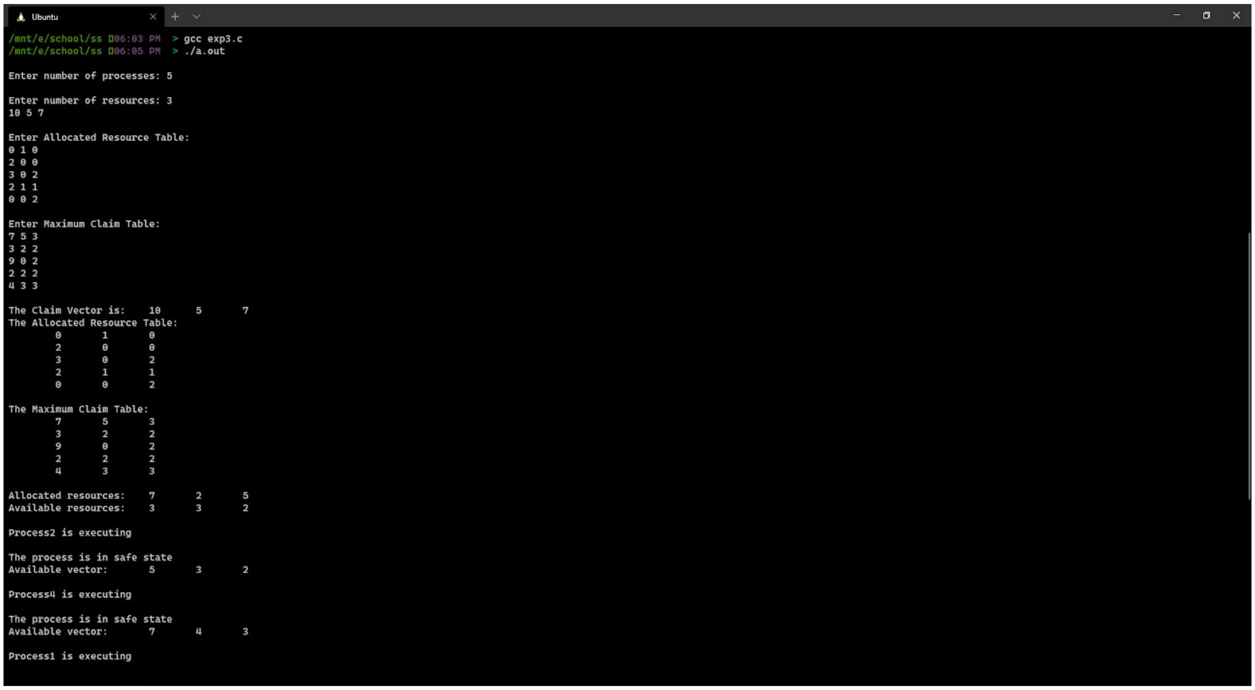
```
    {
        for (j = 0; j < resources; j++)

        {

            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++)

                {
```



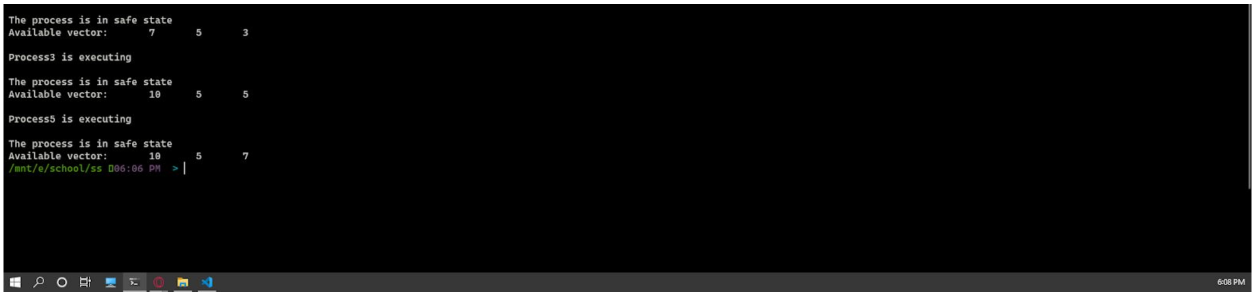
```
                    if (maximum_claim[i][j] - current[i][j] >
available[j])

                    {
                        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:");

```

```c
            for (i = 0; i < resources; i++)

            {

                printf("\t%d", available[i]);
            }

            printf("\n");
        }
    }
    return 0;
}
```

Output



```
/mnt/e/school/ss 06:03 PM  > gcc exp3.c
/mnt/e/school/ss 06:05 PM  > ./a.out

Enter number of processes: 5

Enter number of resources: 3
10 5 7

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

Enter Maximum Claim Table:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3

The Claim Vector is:    10      5       7
The 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
Available vector:       7       4       3

Process1 is executing
```



```
The process is in safe state
Available vector:       7       5       3

Process3 is executing

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
/mnt/e/school/ss 06:06 PM  > |
```