NAME: KATAKAM JASWIN

REGISTRATION NO: 12214571

SECTION: K22ZC

ROLL NO : 66

```
#include <stdio.h>
#include <stdbool.h>

int main() {
    int available[4] = {1, 5, 2, 0};
    int max[5][4] = {
        {0, 0, 1, 2},
        {1, 7, 5, 0},
        {2, 3, 5, 6},
        {0, 6, 5, 2},
        {0, 6, 5, 6}
    };
    int allocation[5][4] = {
        {0, 0, 1, 2},
        {1, 0, 0, 0},
        {2, 3, 5, 6},
        {0, 6, 3, 2},
        {0, 6, 5, 4}
    };
    int need[5][4];

    // Calculate the need matrix
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 4; j++) {
            need[i][j] = max[i][j] - allocation[i][j];
        }
    }
```

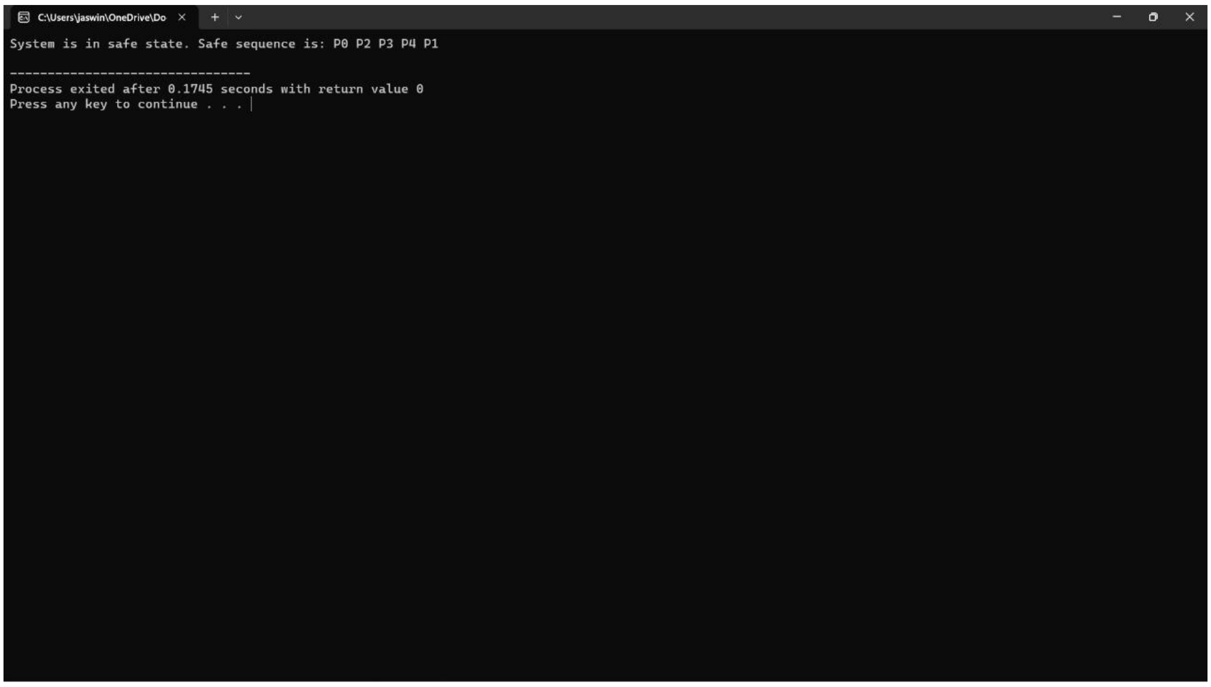
```
bool finish[5] = {false, false, false, false, false};
int work[4];
for (int i = 0; i < 4; i++) {
    work[i] = available[i];
}

int safeSeq[5];
int count = 0;

while (count < 5) {
    bool found = false;
    for (int i = 0; i < 5; i++) {
        if (finish[i] == false) {
            int j;
            for (j = 0; j < 4; j++) {
                if (need[i][j] > work[j]) {
                    break;
                }
            }
            if (j == 4) {
                for (int k = 0; k < 4; k++) {
                    work[k] += allocation[i][k];
                }
                safeSeq[count++] = i;
                finish[i] = true;
                found = true;
            }
        }
    }
    if (found == false) {
            printf("System is not in safe state.\n");
            return 0;
        }
    }

    printf("System is in safe state. Safe sequence is: ");
    for (int i = 0; i < 5; i++) {
        printf("P%d ", safeSeq[i]);
    }
    printf("\n");

    return 0;
}
```



```
System is in safe state. Safe sequence is: P0 P2 P3 P4 P1

--------------------------------
Process exited after 0.1745 seconds with return value 0
Press any key to continue . . .
```

Report:

Banker's Algorithm:

The Banker's algorithm is a deadlock avoidance algorithm used in operating systems. It's designed to ensure that a system can allocate resources to processes in a way that prevents deadlock. The algorithm checks if a sequence of resource requests and releases will keep the system in a safe state.

Available Resources:

The available array represents the number of available instances of each resource type in the system. It's a key part of the Banker's algorithm as it defines the current available resources for allocation.

Max Allocation Matrix:

The max matrix represents the maximum demand of each process for each resource type. It shows the maximum number of resources a process may need. In this code, it's a 5x4 matrix, indicating 5 processes and 4 resource types.

Allocation Matrix:

The allocation matrix represents the number of resources currently allocated to each process. It shows the resources currently in use by each process.

Need Matrix:

The need matrix represents the remaining resource needs for each process. It's calculated by subtracting the allocation matrix from the max matrix. The need matrix shows the resources a process still needs to complete its tasks.

Finish Array:

The finish array is a boolean array that keeps track of whether a process has completed or not. Initially, all processes are marked as "false," and when a process completes, it's marked as "true."

Work Vector:

The work array represents the available resources during the execution of the algorithm. It's initially set to the values in the available array but is modified during the execution of the algorithm.

Safe Sequence:

The safeSeq array stores the safe sequence of processes. If the system is in a safe state, this array will contain a sequence of processes that can be executed without causing a deadlock.