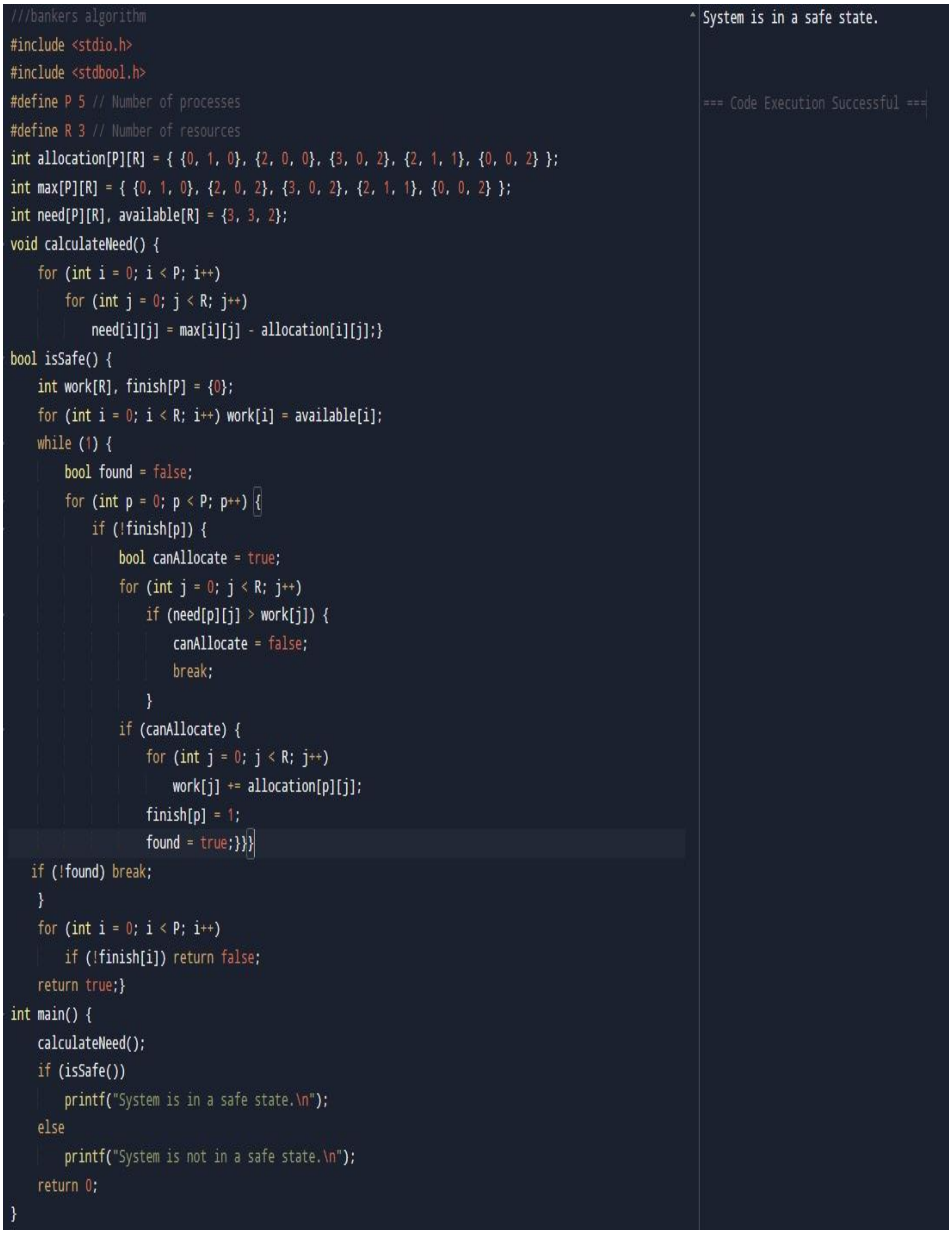
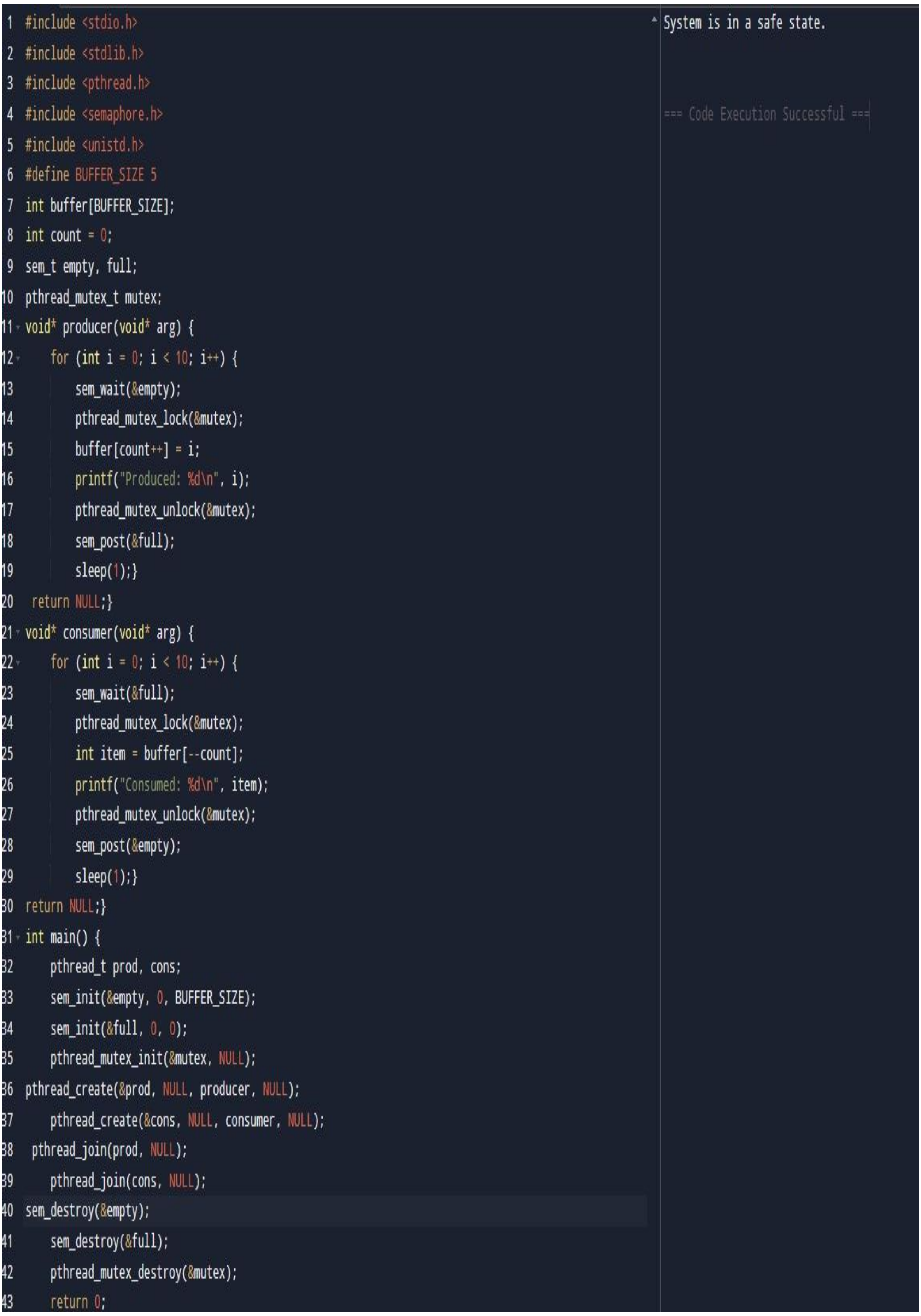


```c
///bankers algorithm
#include <stdio.h>
#include <stdbool.h>
#define P 5 // Number of processes
#define R 3 // Number of resources
int allocation[P][R] = { {0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2} };
int max[P][R] = { {0, 1, 0}, {2, 0, 2}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2} };
int need[P][R], available[R] = {3, 3, 2};
void calculateNeed() {
    for (int i = 0; i < P; i++)
        for (int j = 0; j < R; j++)
            need[i][j] = max[i][j] - allocation[i][j];}
bool isSafe() {
    int work[R], finish[P] = {0};
    for (int i = 0; i < R; i++) work[i] = available[i];
    while (1) {
        bool found = false;
        for (int p = 0; p < P; p++) {
            if (!finish[p]) {
                bool canAllocate = true;
                for (int j = 0; j < R; j++)
                    if (need[p][j] > work[j]) {
                        canAllocate = false;
                        break;
                    }
                if (canAllocate) {
                    for (int j = 0; j < R; j++)
                        work[j] += allocation[p][j];
                    finish[p] = 1;
                    found = true;}}}
   if (!found) break;
    }
    for (int i = 0; i < P; i++)
        if (!finish[i]) return false;
    return true;}
int main() {
    calculateNeed();
    if (isSafe())
        printf("System is in a safe state.\n");
    else
        printf("System is not in a safe state.\n");
    return 0;
}
```

```
System is in a safe state.


=== Code Execution Successful ===
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define BUFFER_SIZE 5
int buffer[BUFFER_SIZE];
int count = 0;
sem_t empty, full;
pthread_mutex_t mutex;
void* producer(void* arg) {
    for (int i = 0; i < 10; i++) {
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[count++] = i;
        printf("Produced: %d\n", i);
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
        sleep(1);}
 return NULL;}
void* consumer(void* arg) {
    for (int i = 0; i < 10; i++) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[--count];
        printf("Consumed: %d\n", item);
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
        sleep(1);}
return NULL;}
int main() {
    pthread_t prod, cons;
    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);
    pthread_mutex_init(&mutex, NULL);
pthread_create(&prod, NULL, producer, NULL);
    pthread_create(&cons, NULL, consumer, NULL);
 pthread_join(prod, NULL);
    pthread_join(cons, NULL);
sem_destroy(&empty);
    sem_destroy(&full);
    pthread_mutex_destroy(&mutex);
    return 0;
```

```
System is in a safe state.


=== Code Execution Successful ===
```