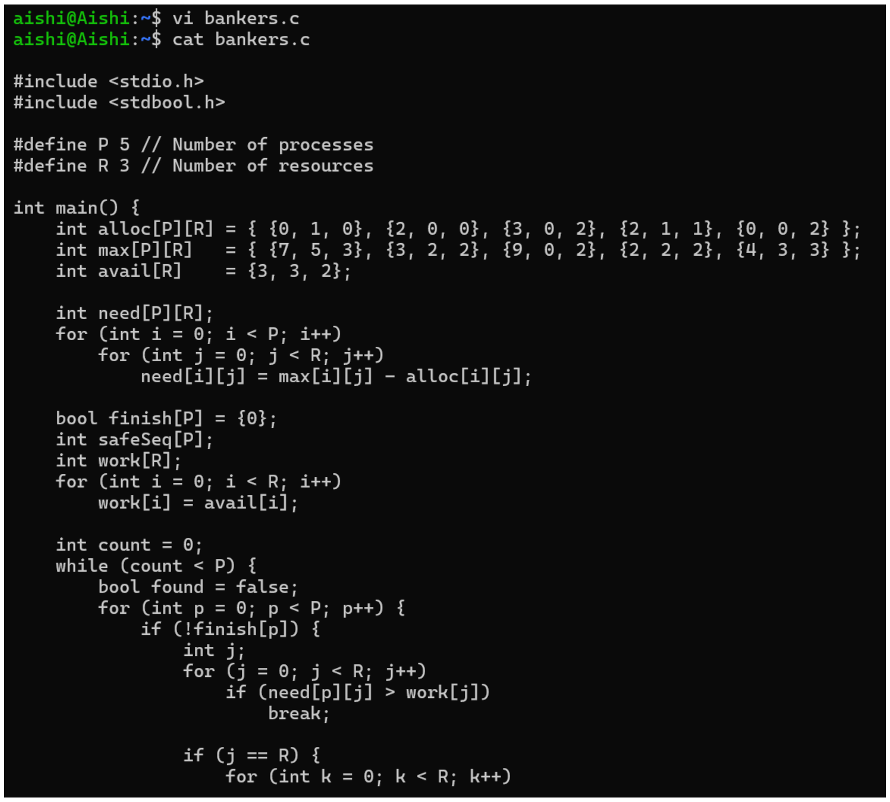
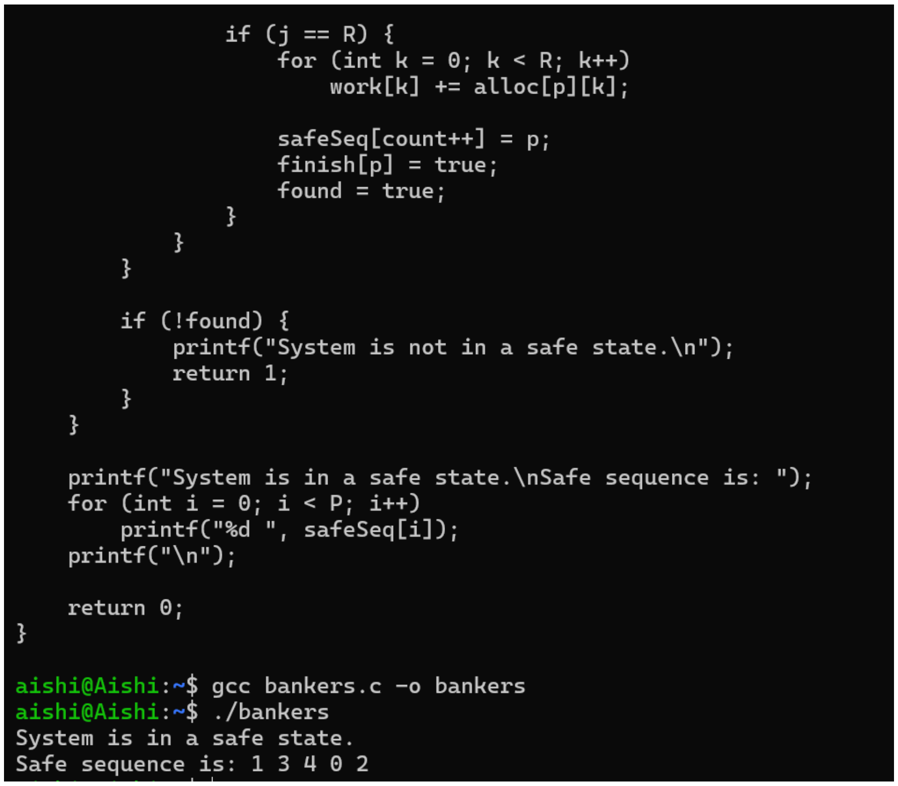
Bankers algorithm

Aishi De



```
aishi@Aishi:~$ vi bankers.c
aishi@Aishi:~$ cat bankers.c

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

#define P 5 // Number of processes
#define R 3 // Number of resources

int main() {
    int alloc[P][R] = { {0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2} };
    int max[P][R]   = { {7, 5, 3}, {3, 2, 2}, {9, 0, 2}, {2, 2, 2}, {4, 3, 3} };
    int avail[R]    = {3, 3, 2};

    int need[P][R];
    for (int i = 0; i < P; i++)
        for (int j = 0; j < R; j++)
            need[i][j] = max[i][j] - alloc[i][j];

    bool finish[P] = {0};
    int safeSeq[P];
    int work[R];
    for (int i = 0; i < R; i++)
        work[i] = avail[i];

    int count = 0;
    while (count < P) {
        bool found = false;
        for (int p = 0; p < P; p++) {
            if (!finish[p]) {
                int j;
                for (j = 0; j < R; j++)
                    if (need[p][j] > work[j])
                        break;

                if (j == R) {
                    for (int k = 0; k < R; k++)
```



```
                if (j == R) {
                    for (int k = 0; k < R; k++)
                        work[k] += alloc[p][k];

                    safeSeq[count++] = p;
                    finish[p] = true;
                    found = true;
                }
            }
        }

        if (!found) {
            printf("System is not in a safe state.\n");
            return 1;
        }
    }

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

    return 0;
}

aishi@Aishi:~$ gcc bankers.c -o bankers
aishi@Aishi:~$ ./bankers
System is in a safe state.
Safe sequence is: 1 3 4 0 2
```