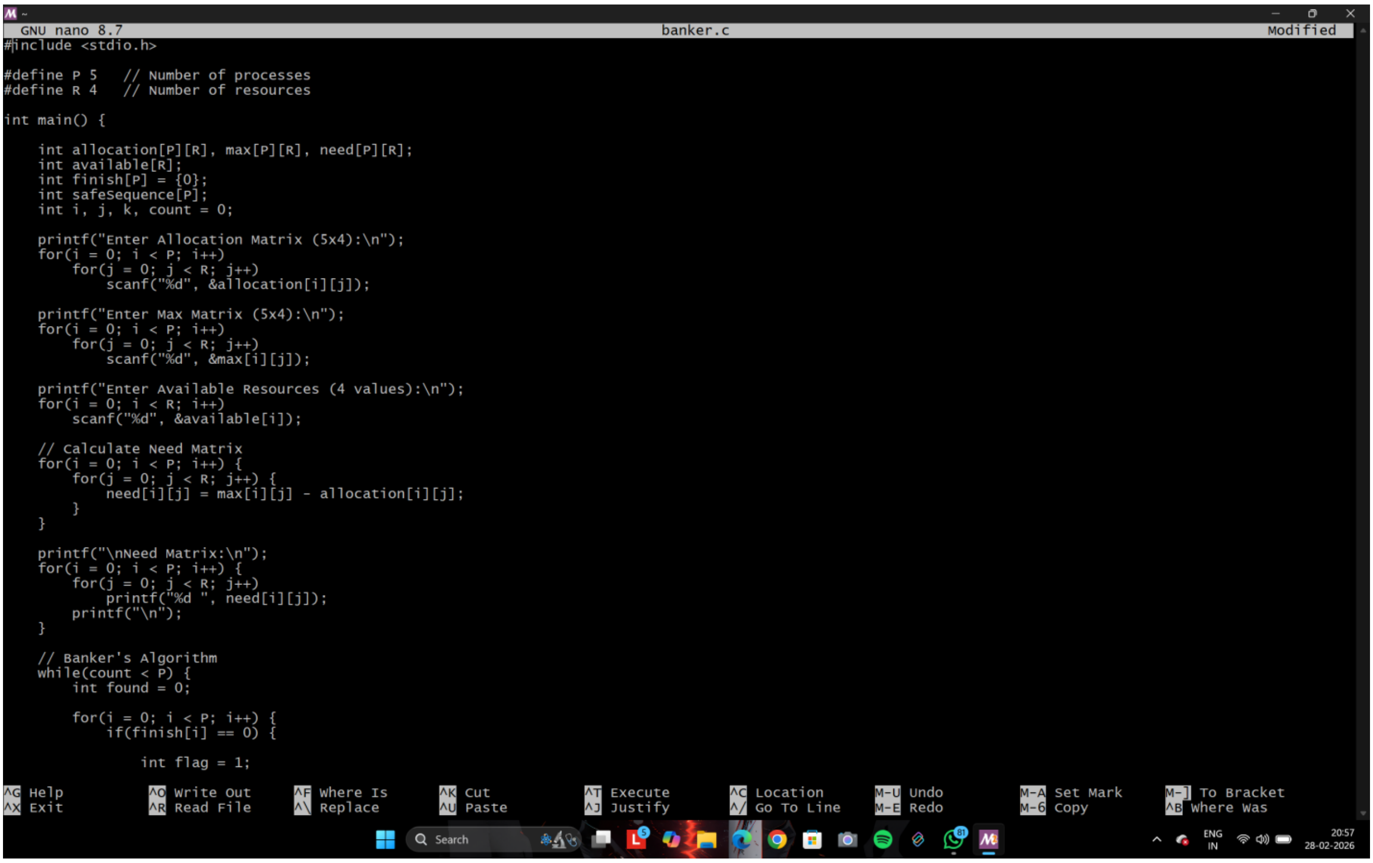
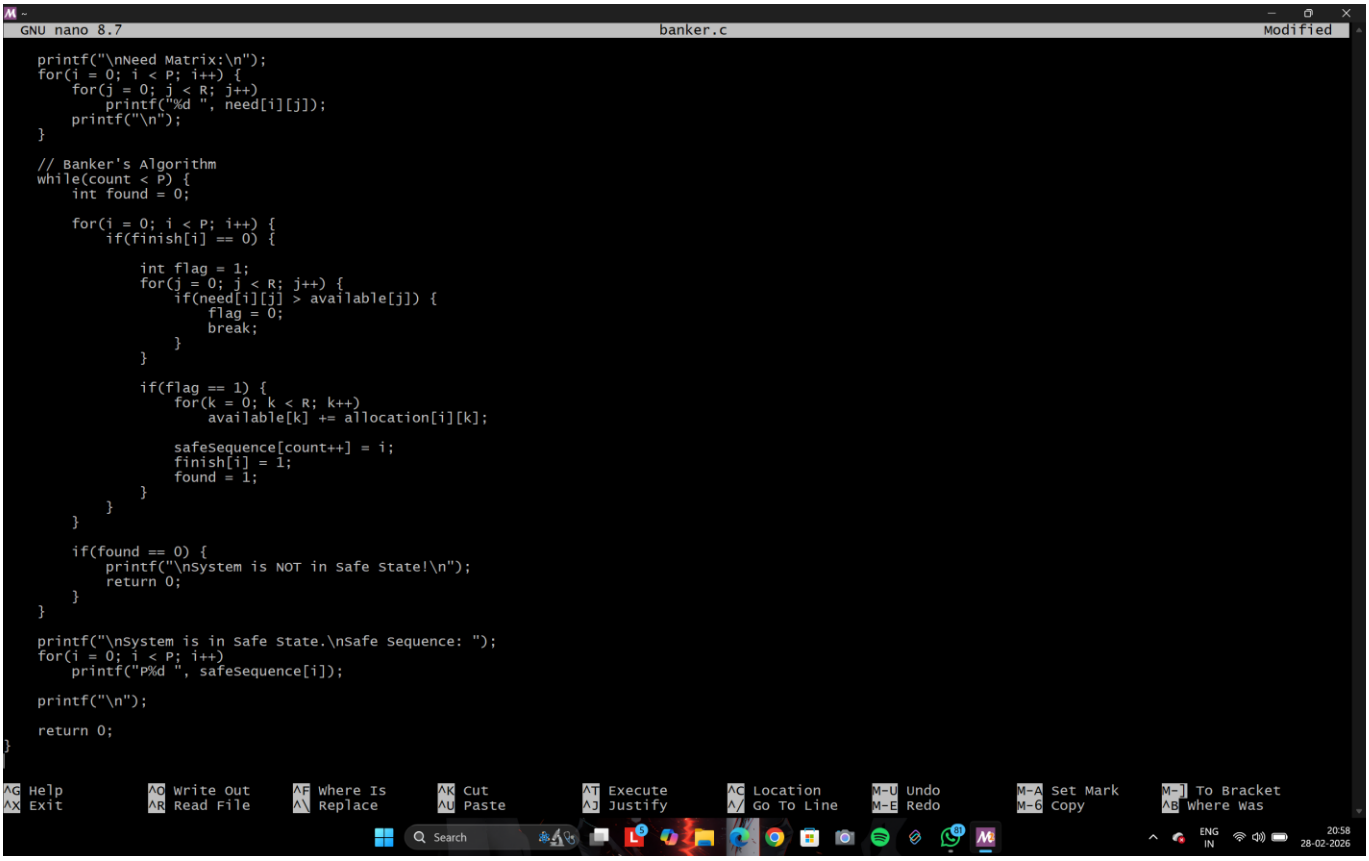
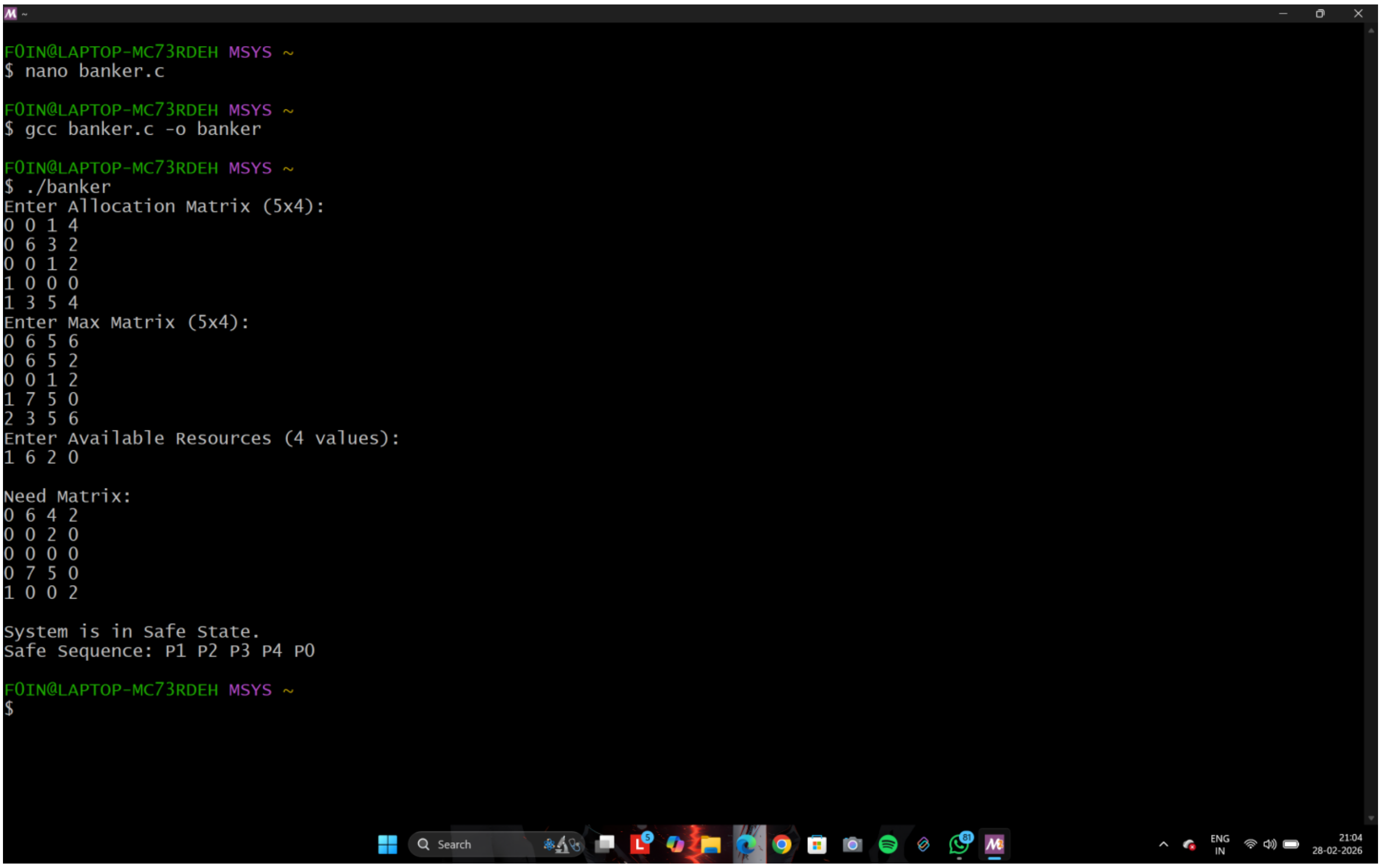


```
#include <stdio.h>

#define P 5   // Number of processes
#define R 4   // Number of resources

int main() {

    int allocation[P][R], max[P][R], need[P][R];
    int available[R];
    int finish[P] = {0};
    int safeSequence[P];
    int i, j, k, count = 0;

    printf("Enter Allocation Matrix (5x4):\n");
    for(i = 0; i < P; i++)
        for(j = 0; j < R; j++)
            scanf("%d", &allocation[i][j]);

    printf("Enter Max Matrix (5x4):\n");
    for(i = 0; i < P; i++)
        for(j = 0; j < R; j++)
            scanf("%d", &max[i][j]);

    printf("Enter Available Resources (4 values):\n");
    for(i = 0; i < R; i++)
        scanf("%d", &available[i]);

    // Calculate Need Matrix
    for(i = 0; i < P; i++) {
        for(j = 0; j < R; j++) {
            need[i][j] = max[i][j] - allocation[i][j];
        }
    }

    printf("\nNeed Matrix:\n");
    for(i = 0; i < P; i++) {
        for(j = 0; j < R; j++)
            printf("%d ", need[i][j]);
        printf("\n");
    }

    // Banker's Algorithm
    while(count < P) {
        int found = 0;

        for(i = 0; i < P; i++) {
            if(finish[i] == 0) {

                int flag = 1;
```



```
    printf("\nNeed Matrix:\n");
    for(i = 0; i < P; i++) {
        for(j = 0; j < R; j++)
            printf("%d ", need[i][j]);
        printf("\n");
    }

    // Banker's Algorithm
    while(count < P) {
        int found = 0;

        for(i = 0; i < P; i++) {
            if(finish[i] == 0) {

                int flag = 1;
                for(j = 0; j < R; j++) {
                    if(need[i][j] > available[j]) {
                        flag = 0;
                        break;
                    }
                }

                if(flag == 1) {
                    for(k = 0; k < R; k++)
                        available[k] += allocation[i][k];

                    safeSequence[count++] = i;
                    finish[i] = 1;
                    found = 1;
                }
            }
        }

        if(found == 0) {
            printf("\nSystem is NOT in Safe State!\n");
            return 0;
        }
    }

    printf("\nSystem is in Safe State.\nSafe Sequence: ");
    for(i = 0; i < P; i++)
        printf("P%d ", safeSequence[i]);

    printf("\n");

    return 0;
}
```



```
F0IN@LAPTOP-MC73RDEH MSYS ~
$ nano banker.c

F0IN@LAPTOP-MC73RDEH MSYS ~
$ gcc banker.c -o banker

F0IN@LAPTOP-MC73RDEH MSYS ~
$ ./banker
Enter Allocation Matrix (5x4):
0 0 1 4
0 6 3 2
0 0 1 2
1 0 0 0
1 3 5 4
Enter Max Matrix (5x4):
0 6 5 6
0 6 5 2
0 0 1 2
1 7 5 0
2 3 5 6
Enter Available Resources (4 values):
1 6 2 0

Need Matrix:
0 6 4 2
0 0 2 0
0 0 0 0
0 7 5 0
1 0 0 2

System is in Safe State.
Safe Sequence: P1 P2 P3 P4 P0

F0IN@LAPTOP-MC73RDEH MSYS ~
$
```