S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT & RESEARCH, NAGPUR

Practical 07

Aim: Develop a program to manage resource allocation for five processes (Google Drive, Firefox, Word Processor, Excel, and PowerPoint) using four types of resources (Printer, ROM, Hard Disk, and RAM). The program takes input for allocated, maximum, and available resources, calculates the current need of each process, and determines if a safe execution order exists using the Banker's Algorithm.

Name:

USN:

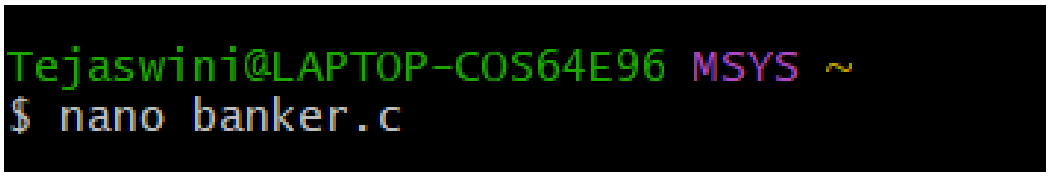
Semester / Year: 4th SEM / 2nd YEAR

Academic Session: 2025-26

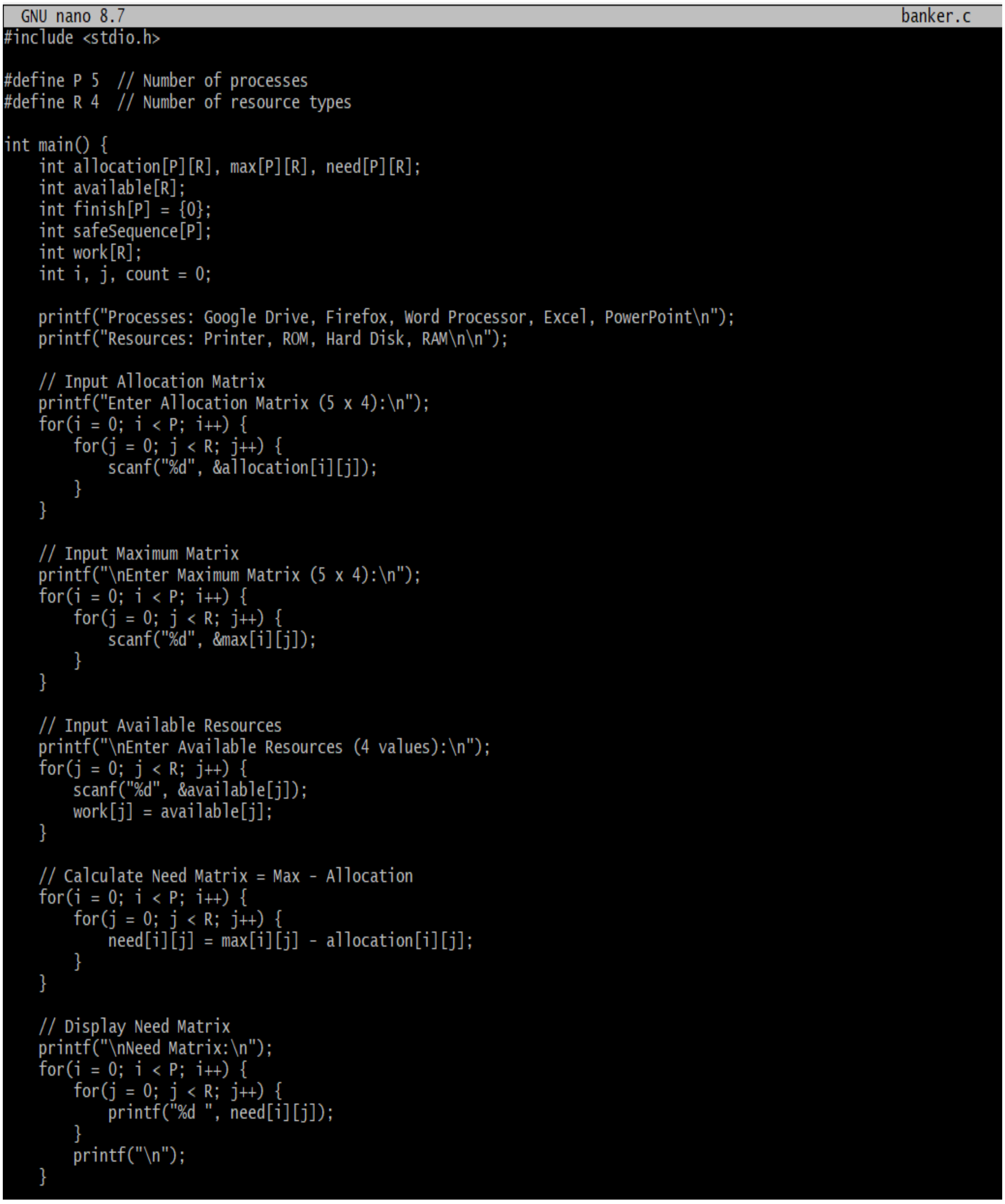
Date of Performance:

Date of Submission:

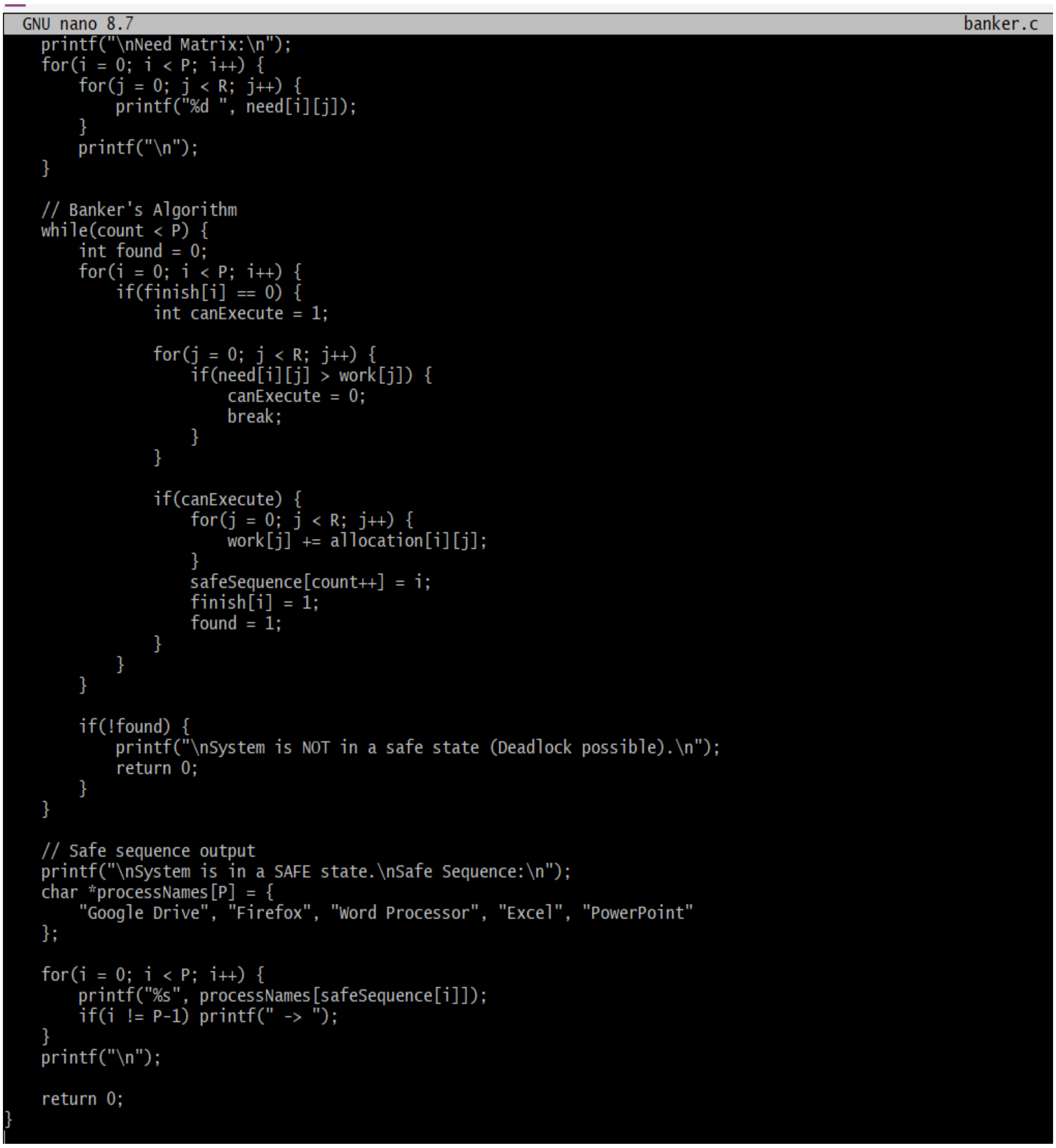
❖ CODE :



```
Tejaswini@LAPTOP-COS64E96 MSYS ~
$ nano banker.c
```

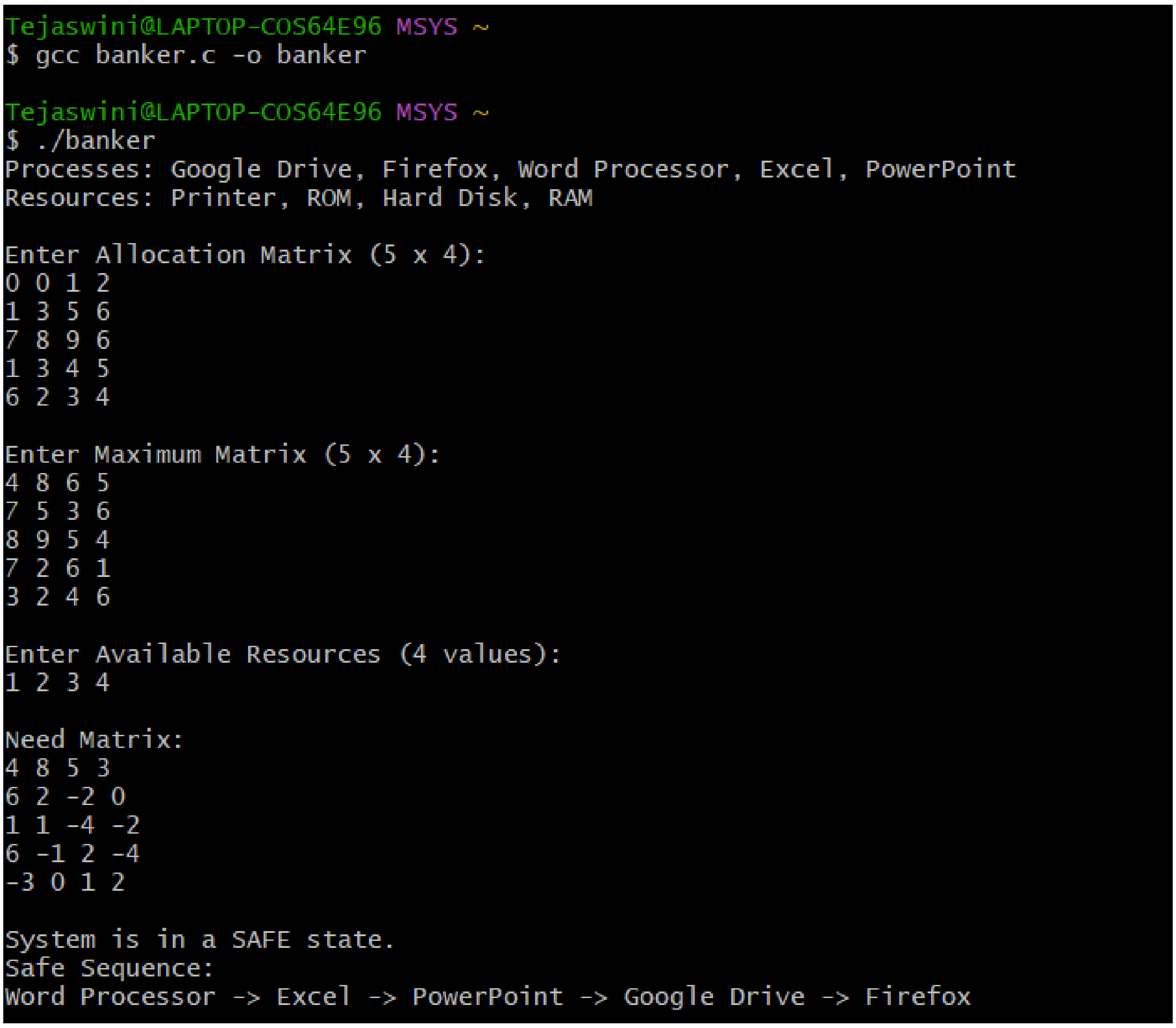


```
  GNU nano 8.7                                                                                                banker.c
#include <stdio.h>

#define P 5  // Number of processes
#define R 4  // Number of resource types

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

    printf("Processes: Google Drive, Firefox, Word Processor, Excel, PowerPoint\n");
    printf("Resources: Printer, ROM, Hard Disk, RAM\n\n");

    // Input Allocation Matrix
    printf("Enter Allocation Matrix (5 x 4):\n");
    for(i = 0; i < P; i++) {
        for(j = 0; j < R; j++) {
            scanf("%d", &allocation[i][j]);
        }
    }

    // Input Maximum Matrix
    printf("\nEnter Maximum Matrix (5 x 4):\n");
    for(i = 0; i < P; i++) {
        for(j = 0; j < R; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    // Input Available Resources
    printf("\nEnter Available Resources (4 values):\n");
    for(j = 0; j < R; j++) {
        scanf("%d", &available[j]);
        work[j] = available[j];
    }

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

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



```
  GNU nano 8.7                                                                                   banker.c
    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 canExecute = 1;

                for(j = 0; j < R; j++) {
                    if(need[i][j] > work[j]) {
                        canExecute = 0;
                        break;
                    }
                }

                if(canExecute) {
                    for(j = 0; j < R; j++) {
                        work[j] += allocation[i][j];
                    }
                    safeSequence[count++] = i;
                    finish[i] = 1;
                    found = 1;
                }
            }
        }

        if(!found) {
            printf("\nSystem is NOT in a safe state (Deadlock possible).\n");
            return 0;
        }
    }

    // Safe sequence output
    printf("\nSystem is in a SAFE state.\nSafe Sequence:\n");
    char *processNames[P] = {
        "Google Drive", "Firefox", "Word Processor", "Excel", "PowerPoint"
    };

    for(i = 0; i < P; i++) {
        printf("%s", processNames[safeSequence[i]]);
        if(i != P-1) printf(" -> ");
    }
    printf("\n");

    return 0;
}
```

❖ OUTPUT :



```
Tejaswini@LAPTOP-COS64E96 MSYS ~
$ gcc banker.c -o banker

Tejaswini@LAPTOP-COS64E96 MSYS ~
$ ./banker
Processes: Google Drive, Firefox, Word Processor, Excel, PowerPoint
Resources: Printer, ROM, Hard Disk, RAM

Enter Allocation Matrix (5 x 4):
0 0 1 2
1 3 5 6
7 8 9 6
1 3 4 5
6 2 3 4

Enter Maximum Matrix (5 x 4):
4 8 6 5
7 5 3 6
8 9 5 4
7 2 6 1
3 2 4 6

Enter Available Resources (4 values):
1 2 3 4

Need Matrix:
4 8 5 3
6 2 -2 0
1 1 -4 -2
6 -1 2 -4
-3 0 1 2

System is in a SAFE state.
Safe Sequence:
Word Processor -> Excel -> PowerPoint -> Google Drive -> Firefox
```