Name : Sujal Santosh Porte
PRN No : 122B1B227

Assignment No. 05

Problem Statement : Write a program to implement Banker's Algorithm for deadlock avoidance.

Code:

safe.c

```
#include <stdio.h>

void bankersAlgorithm() {
    // P0, P1, P2, P3, P4 are the Process names here
    int n, m, i, j, k;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter the number of resources: ");
    scanf("%d", &m);

    int alloc[n][m];
    int max[n][m];
    int avail[m];
    printf("Enter allocation Matrix (%d x %d):\n", n, m);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &alloc[i][j]);
        }
    }

    printf("Enter MAX Matrix (%d x %d):\n", n, m);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    printf("Enter Available Resources (%d values):\n", m);
    for (int i = 0; i < m; i++) { scanf("%d", &avail[i]);

    }
```

```
int f[n], ans[n], ind = 0;

for (k = 0; k < n; k++) {
    f[k] = 0;
}

int need[n][m];
for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++)
        need[i][j] = max[i][j] - alloc[i][j];
}

int y = 0;
for (k = 0; k < 5; k++) {
    for (i = 0; i < n; i++) {
        if (f[i] == 0) {
            int flag = 0;
            for (j = 0; j < m; j++) {
                if (need[i][j] > avail[j]) {
                    flag = 1;
                    break;
                }
            }
            if (flag == 0) {
                ans[ind++] = i;
                for (y = 0; y < m; y++)
                    avail[y] += alloc[i][y];
                f[i] = 1;
            }
        }
    }
}

int flag = 1;
for (i = 0; i < n; i++) {
    if (f[i] == 0) {
        flag = 0;
        printf("The following system is not safe\n");
        return;
    }
}

if (flag == 1) {
    printf("Following is the SAFE Sequence:\n");
```

```
        for (i = 0; i < n - 1; i++)
            printf(" P%d ->", ans[i]);
        printf(" P%d\n", ans[n - 1]);
    }
}
```

Request.c

```
#include <stdio.h>
void request() {

    int n , m;

    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter the number of resources: ");
    scanf("%d", &m);

    int alloc[n][m];
    int max[n][m];
    int avail[m];
    printf("Enter allocation Matrix (%d x %d):\n", n, m);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &alloc[i][j]);
        }
    }

    printf("Enter MAX Matrix (%d x %d):\n", n, m);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    printf("Enter Available Resources (%d values):\n", m);
    for (int i = 0; i < m; i++) { scanf("%d", &avail[i]);

    }

    int need[n][m], f[n], ans[n], ind = 0;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
```

```
            need[i][j] = max[i][j] - alloc[i][j];
        }
    }

    int process;
    printf("Enter process number (0 to %d) making the request: ", n - 1);
    scanf("%d", &process);


    int request[m];
    printf("Enter resource request for P%d: ", process);
    for (int i = 0; i < m; i++) {
        scanf("%d", &request[i]);
    }

    for (int i = 0; i < m; i++) {
        if (request[i] > need[process][i]) {
            printf("\nProcess P%d requested more than its maximum need. Request denied.\n",
process);
            return;
        }
    }


    for (int i = 0; i < m; i++) {
        if (request[i] > avail[i]) {
            printf("\nNot enough resources available. Request denied.\n");
            return;
        }
    }


    for (int i = 0; i < m; i++) {
        avail[i] -= request[i];
        alloc[process][i] += request[i];
        need[process][i] -= request[i];
    }


    for (int k = 0; k < n; k++) f[k] = 0;
    ind = 0;

    for (int k = 0; k < n; k++) {
        for (int i = 0; i < n; i++) {
```

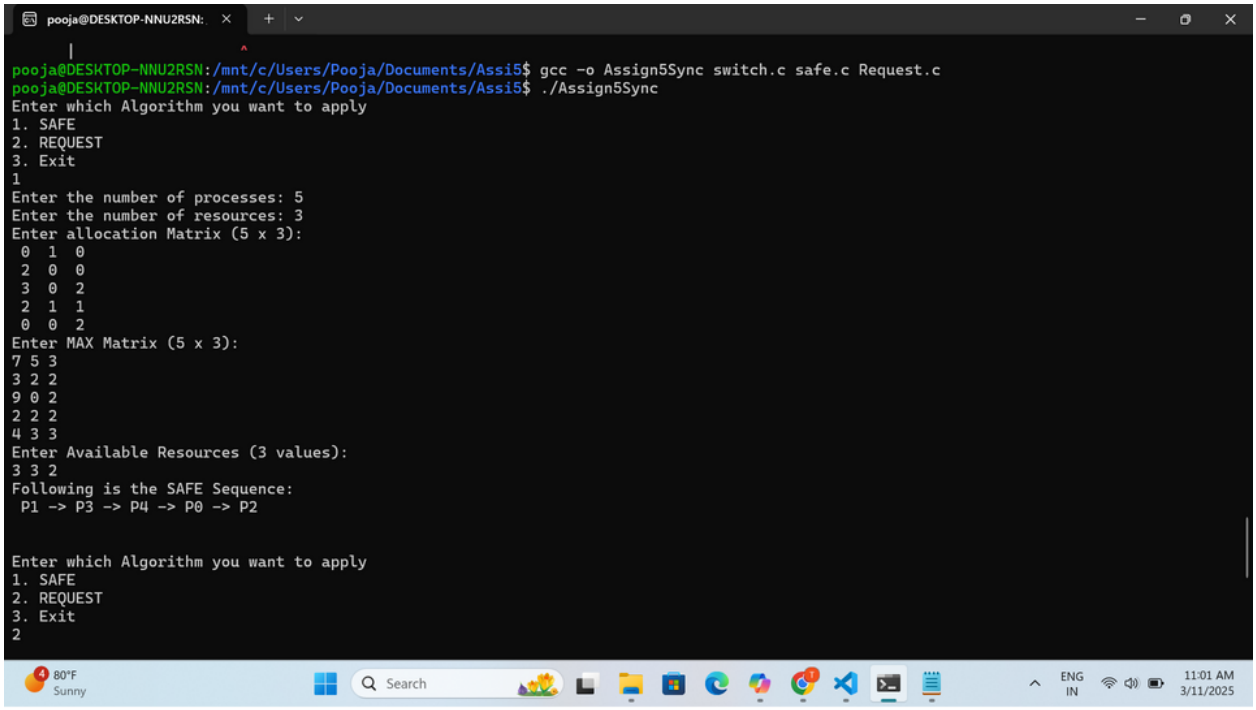
```
            if (f[i] == 0) {
                int flag = 0;
                for (int j = 0; j < m; j++) {
                    if (need[i][j] > avail[j]) {
                        flag = 1;
                        break;
                    }
                }
                if (flag == 0) {
                    ans[ind++] = i;
                    for (int y = 0; y < m; y++)
                        avail[y] += alloc[i][y];
                    f[i] = 1;
                }
            }
        }
    }

    int flag = 1;
    for (int i = 0; i < n; i++) {
        if (f[i] == 0) {
            flag = 0;
            printf("\nThe request leads to an unsafe state. Request denied.\n");
            return;
        }
    }

    printf("\nThe request is granted. System remains in a SAFE state.\nSafe Sequence: ");
    for (int i = 0; i < n - 1; i++)
        printf(" P%d ->", ans[i]);
     printf(" P%d\n", ans[n - 1]);
}
```

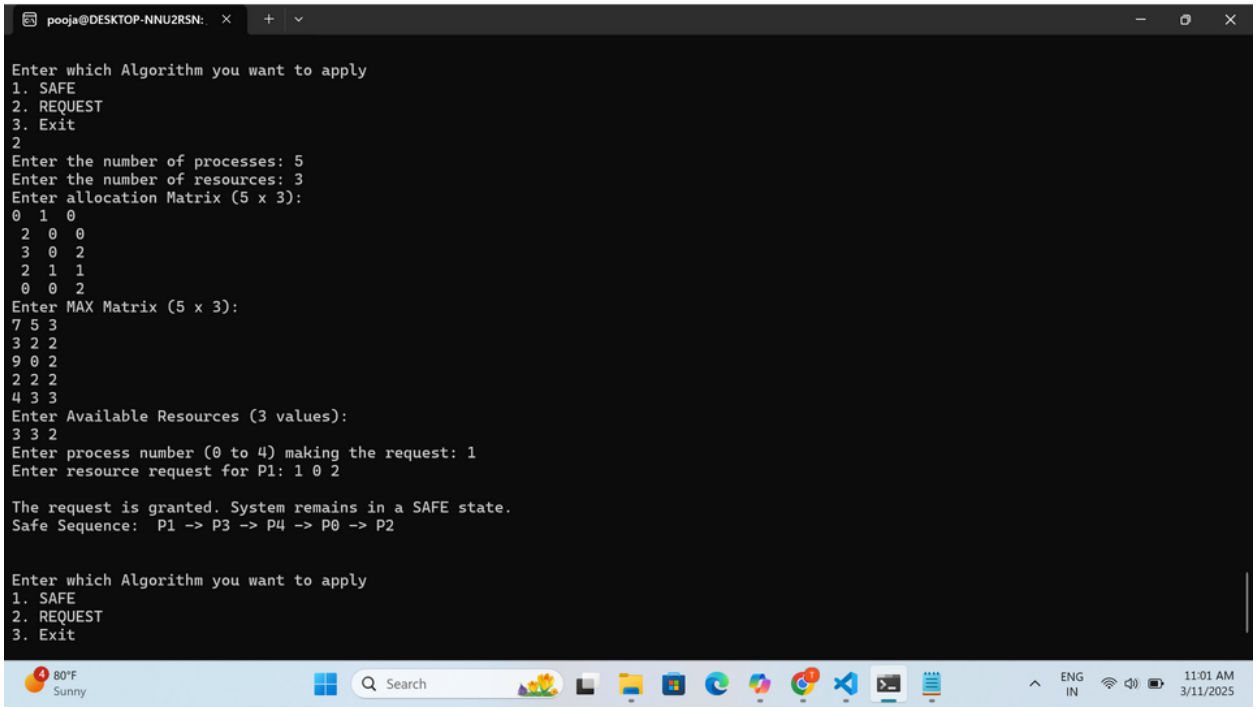
switch.c

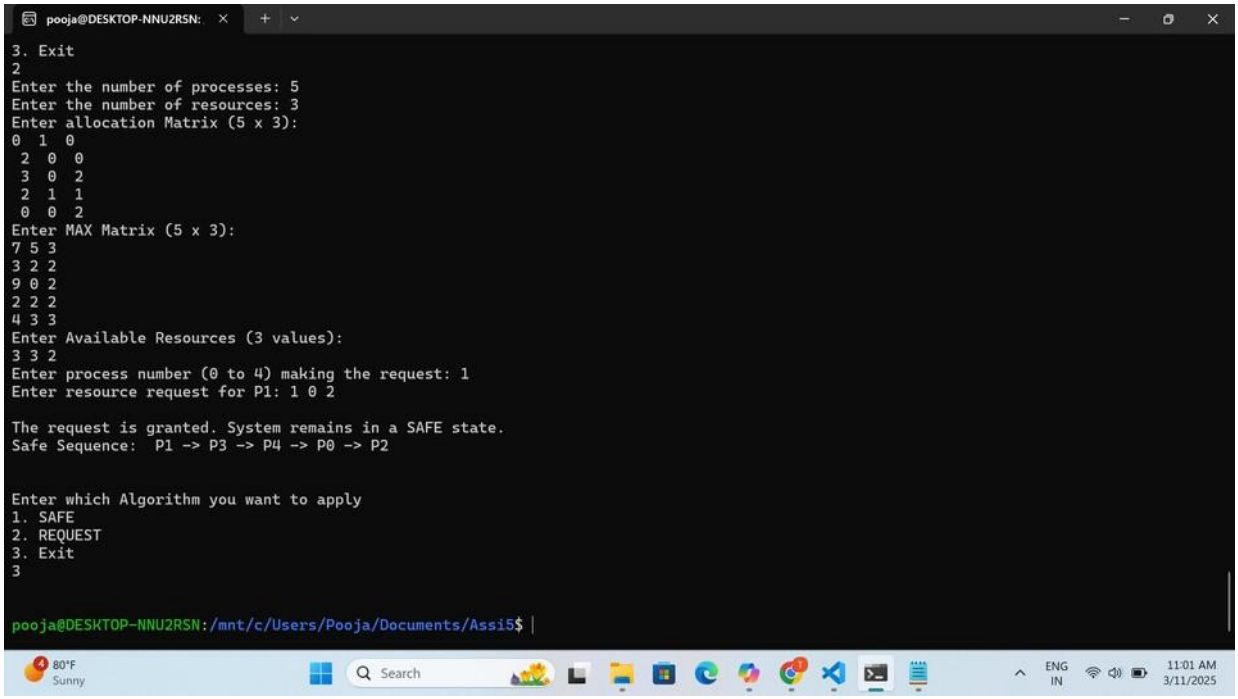
```
#include  <stdio.h>
#include <stdlib.h>
#include <limits.h>
// Function Declarations
void bankersAlgorithm();
void request();
```

```
void main(){

        int choice;
        int flag =1;

        while(flag){
                printf("Enter which Algorithm you want to apply\n");
                printf("1. SAFE \n2. REQUEST\n3. Exit\n");
                scanf("%d", &choice);


                switch(choice){
                        case 1:
                                bankersAlgorithm();
                                        break;
                        case 2:
                                request();
                                        break;

                        case 3:
                                        flag=0;
                                        break;
                        default :
                                        printf("Invalid choice");
                                        break;
                }
                printf("\n\n");
        }
}
```

OUTPUT:


pooja@DESKTOP-NNU2RSN:
pooja@DESKTOP-NNU2RSN:/mnt/c/Users/Pooja/Documents/Assi5$ gcc -o Assign5Sync switch.c safe.c Request.c
pooja@DESKTOP-NNU2RSN:/mnt/c/Users/Pooja/Documents/Assi5$ ./Assign5Sync
Enter which Algorithm you want to apply
1. SAFE
2. REQUEST
3. Exit
1
Enter the number of processes: 5
Enter the number of resources: 3
Enter allocation Matrix (5 x 3):
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter MAX Matrix (5 x 3):
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter Available Resources (3 values):
3 3 2
Following is the SAFE Sequence:
P1 -> P3 -> P4 -> P0 -> P2
Enter which Algorithm you want to apply
1. SAFE
2. REQUEST
3. Exit
2
80°F
Sunny
Search
ENG
IN
11:01 AM
3/11/2025


pooja@DESKTOP-NNU2RSN:
Enter which Algorithm you want to apply
1. SAFE
2. REQUEST
3. Exit
2
Enter the number of processes: 5
Enter the number of resources: 3
Enter allocation Matrix (5 x 3):
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter MAX Matrix (5 x 3):
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter Available Resources (3 values):
3 3 2
Enter process number (0 to 4) making the request: 1
Enter resource request for P1: 1 0 2
The request is granted. System remains in a SAFE state.
Safe Sequence: P1 -> P3 -> P4 -> P0 -> P2
Enter which Algorithm you want to apply
1. SAFE
2. REQUEST
3. Exit
80°F
Sunny
Search
ENG
IN
11:01 AM
3/11/2025


pooja@DESKTOP-NNU2RSN:
3. Exit
2
Enter the number of processes: 5
Enter the number of resources: 3
Enter allocation Matrix (5 x 3):
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter MAX Matrix (5 x 3):
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter Available Resources (3 values):
3 3 2
Enter process number (0 to 4) making the request: 1
Enter resource request for P1: 1 0 2
The request is granted. System remains in a SAFE state.
Safe Sequence: P1 -> P3 -> P4 -> P0 -> P2
Enter which Algorithm you want to apply
1. SAFE
2. REQUEST
3. Exit
3
pooja@DESKTOP-NNU2RSN:/mnt/c/Users/Pooja/Documents/Assi5$
80°F
Sunny
Search
ENG
IN
11:01 AM
3/11/2025