

Banker's Algorithm OS Lab

:- By 1BM21CS232

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int N, M = 3, ind = 0;
    printf("\nEnter the number of processes: ");
    scanf("%d", &N);

    int alloc[N][M], max[N][M], need[N][M], finished[N],
    ans[N], avail[M];

    printf("\nEnter allocated resources\n");
    for (int i = 0; i < N; i++)
    {
        printf("For Process %d: ", i);
        for (int j = 0; j < M; j++)
        {
            scanf("%d", &alloc[i][j]);
        }
    }

    printf("\nEnter Maximum resources\n");
    for (int i = 0; i < N; i++)
    {
        printf("For Process %d: ", i);
        for (int j = 0; j < M; j++)
```

```
    {  
        scanf("%d", &max[i][j]);  
    }  
}
```

```
printf("\nEnter available resources\n");  
for (int i = 0; i < M; i++)  
{  
    scanf("%d",&avail[i]);  
}
```

```
for (int i = 0; i < N; i++)  
{  
    finished[i] = 0;  
}
```

```
for (int i = 0; i < N; i++)  
{  
    for (int j = 0; j < M; j++)  
    {  
        need[i][j] = max[i][j] - alloc[i][j];  
    }  
}
```

```
for (int k = 0; k < N; k++)  
{  
    for (int i = 0; i < N; i++)  
    {  
        if (finished[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 p = 0; p < M; p++)
    {
        avail[p] += alloc[i][p];
    }
    finished[i] = 1;
}
}
}
}

```

```

int flag = 1;
for (int i = 0; i < N; i++)
{
    if (finished[i] == 0)
    {
        flag = 0;
        printf("The System is NOT safe\n");
    }
}

```

```

        break;
    }
}

if (flag == 1)
{
    printf("\nSafe Sequence:\n");
    for (int i = 0; i < N - 1; i++)
    {
        printf("P%d --> ", ans[i]);
    }
    printf("P%d\n", ans[N - 1]);
}
}

```

Output:

The screenshot shows the Code-Blocks IDE with the following components:

- Code Editor:** Displays the C code for the Banker's Algorithm. The code includes a loop to check for a safe sequence and a function to print the sequence.
- Output Window:** Shows the program's execution. It prompts the user to enter the number of processes (5), allocated resources for each process, maximum resources, and available resources. The program then outputs a safe sequence of processes: P1 --> P3 --> P4 --> P0 --> P2.
- Log & Messages Window:** Shows the build process, including the compilation of the program and the execution time (97.869 s).

The output window displays the following text:

```

Enter the number of processes: 5
Enter allocated resources
For Process 0: 0 1 0
For Process 1: 2 0 0
For Process 2: 3 0 2
For Process 3: 2 1 1
For Process 4: 0 0 2
Enter Maximum resources
For Process 0: 7 5 3
For Process 1: 3 2 2
For Process 2: 9 0 2
For Process 3: 2 2 2
For Process 4: 4 3 3
Enter available resources
3 3 2
Safe Sequence:
P1 --> P3 --> P4 --> P0 --> P2
Process returned 0 (0x0)   execution time : 97.869 s
Press any key to continue.

```