

AIM:-

A computer system has four different types of resources (Printer, ROM, hard disk and RAM) and it needs to complete execution of five processes (Google Drive, Firefox, Word Processor, Excel and PowerPoint). The number of allocated resources, maximum needed resources to complete the execution and total available resources should be taken from the user. (Sample data for reference is given below)

Process	Allocation				MAX			
	Printer	ROM	Hard Disk	RAM	Printer	ROM	Hard Disk	RAM
Google Drive	0	0	1	4	0	6	5	6
Firefox	0	6	3	2	0	6	5	2
Word Processor	0	0	1	2	0	0	1	2
Excel	1	0	0	0	1	7	5	0
PowerPoint	1	3	5	4	2	3	5	6

Available Resources			
Printer	ROM	Hard Disk	RAM
1	6	2	0

Write a program to :

- Calculate current need of resources by each process.
- Find whether these processes can be executed with available resources. If yes, show the correct order of process execution.

CODE:-

```
M- GNU nano 8.7                                         BankersAlgo.c
#include <stdio.h>

#define P 5
#define R 4

int main() {
    int allocation[P][R] = {
        {0, 0, 1, 4}, // Google Drive
        {0, 6, 3, 2}, // Firefox
        {0, 0, 1, 2}, // Word Processor
        {1, 0, 0, 0}, // Excel
        {1, 3, 5, 4} // PowerPoint
    };

    int max[P][R] = {
        {0, 6, 5, 6}, // Google Drive
        {0, 6, 5, 2}, // Firefox
        {0, 0, 1, 2}, // Word Processor
        {1, 7, 5, 0}, // Excel
        {2, 3, 5, 6} // PowerPoint
    };

    int available[R] = {1, 6, 2, 0};

    ^G Help      ^O Write Out   ^F Where Is   ^K Cut       ^T Execute
    ^X Exit      ^R Read File   ^X Replace   ^U Paste     ^J Justify
```

M- GNU nano 8.7 BankersAlgo.c

```
int available[R] = {1, 6, 2, 0};

int need[P][R];
int finish[P] = {0};
int safeSeq[P];
int work[R];

char *processes[P] = {
    "Google Drive",
    "Firefox",
    "Word Processor",
    "Excel",
    "PowerPoint"
};

int i, j, count = 0;

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

// Copy available to work
for(j = 0; j < R; j++)
    work[j] = available[j];

// 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] > work[j]) {
                    flag = 0;
                    break;
                }
            }
        }
    }
}
```

AG Help **AO Write Out** **AF Where Is** **AK Cut** **AT Execute**
AX Exit **AR Read File** **AM Replace** **AU Paste** **AJ Justify**

M- GNU nano 8.7 BankersAlgo.c

```
need[i][j] = max[i][j] - allocation[i][j];
printf("%d ", need[i][j]);
}
printf("\n");

// Copy available to work
for(j = 0; j < R; j++)
    work[j] = available[j];

// 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] > work[j]) {
                    flag = 0;
                    break;
                }
            }
        }
    }
}
```

AG Help **AO Write Out** **AF Where Is** **AK Cut** **AT Execute**
AX Exit **AR Read File** **AM Replace** **AU Paste** **AJ Justify**

```
M - GNU nano 8.7                                     BankersAlgo.c
        break;
    }
}

if(flag) {
    for(j = 0; j < R; j++)
        work[j] += allocation[i][j];
    safeSeq[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:\n");
for(i = 0; i < P; i++)
    printf("%s -> ", processes[safeSeq[i]]);

printf("END\n");
return 0;
}

^G Help      ^O Write Out     ^F Where Is      ^K Cut          ^T Execute
^X Exit      ^R Read File     ^V Replace      ^U Paste         ^J Justify
```

OUTPUT:-

```
M -
user@AnushkaGudadhe MSYS ~
$ nano BankersAlgo.c

user@AnushkaGudadhe MSYS ~
$ gcc BankersAlgo.c -o BankersAlgo

user@AnushkaGudadhe MSYS ~
$ ./BankersAlgo

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:
Firefox -> Word Processor -> Excel -> PowerPoint -> Google Drive

user@AnushkaGudadhe MSYS ~
$ |
```

➤ **Given Data**

Available Resources

Printer = 1

ROM = 6

Hard Disk = 2

RAM = 0

➤ **Need Matrix (Max – Allocation)**

Process	Printer	ROM	Hard Disk	RAM
Google Drive	0	6	4	2
Firefox	0	0	2	0
Word Processor	0	0	0	0
Excel	0	7	5	0
PowerPoint	1	0	0	2

➤ **Step-by-Step Execution Check**

Initial Available:

(1, 6, 2, 0)

1. Word Processor

Need = (0,0,0,0) ✓ Can execute

New Available =

(1,6,2,0) + Allocation(0,0,1,2)

= (1,6,3,2)

2. Firefox

Need = (0,0,2,0) ✓ Can execute

New Available =
 $(1,6,3,2) + \text{Allocation}(0,6,3,2) =$
 $(1,12,6,4)$

3.Google Drive

Need = $(0,6,4,2)$ ✓ Can execute

New Available =
 $(1,12,6,4) + \text{Allocation}(0,0,1,4) =$
 $(1,12,7,8)$

4.Excel

Need = $(0,7,5,0)$ ✓ Can execute

New Available =
 $(1,12,7,8) + \text{Allocation}(1,0,0,0) =$
 $(2,12,7,8)$

5.PowerPoint

Need = $(1,0,0,2)$ ✓ Can execute

New Available =
 $(2,12,7,8) + \text{Allocation}(1,3,5,4) =$
 $(3,15,12,12)$

➤ Final Result

✓ YES, all processes can be executed.

✓ The system is in Safe State.

➤ Safe Sequence

Firefox -> Word Processor -> Excel -> PowerPoint -> Google Drive ->

END