



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

Academic Year: 2024-25

Semester: IV

Class / Branch: SE AIML

Subject: CSL403 Operating System Lab

Name of Instructor: Prof. Poonam Tiware

Name of Student: Pandey Kalash

Student ID:23106049

Date of Performance:28-02-25

Date of Submission:28-02-25

Experiment No.6

Aim: Write a program to demonstrate the concept of deadlock avoidance through Banker's Algorithm.

Program 1:

```
1 #include <stdio.h>
2 int main() {
3     int max[5] = {10, 15, 8, 5, 12};
4     int alloc[5] = {5, 7, 4, 2, 6};
5     int need[5];
6     int finish[5] = {0, 0, 0, 0, 0};
7     int safe_sequence[5];
8     int available = 3;
9     for (int i = 0; i < 5; i++) {
10        need[i] = max[i] - alloc[i];
11    }
12    for (int i = 0; i < 5; i++) {
13        for (int j = 0; j < 5; j++) {
14            if (finish[j]) {
15                continue;
16            }
17            else if (need[j] <= available) {
18                available -= alloc[j];
19                finish[j] = 1;
20                safe_sequence[i] = j + 1;
21                printf("Process %d is finished\n", j + 1);
22                break;
23            }
24        }
25    }
26    printf("The safe sequence is \n");
27    for (int i = 0; i < 5; i++) {
28        printf("P%d ", safe_sequence[i]);
29    }
30    printf("\n");
31    return 0;
32 }
```



Output 1:

```
20 + 100%  
REB BOTS REPORT BOTS  
Programiz PRO  
Output Clear  
Process 4 is finished  
Process 1 is finished  
Process 2 is finished  
Process 3 is finished  
Process 5 is finished  
The safe sequence is  
P4 P1 P2 P3 P5  
=== Code Execution Successful ===  
AD  
Programiz PRO  
Premium  
Courses by  
Programiz  
Learn More
```

Program 2:

```
C  
1 #include <stdio.h>  
2  
3 int main() {  
4     int n = 5;  
5     int totalResources = 3;  
6  
7     int alloc[5][3] = {  
8         { 0, 1, 0 },  
9         { 2, 0, 0 },  
10        { 3, 0, 2 },  
11        { 2, 1, 1 },  
12        { 0, 0, 2 }  
13    };  
14  
15    int max[5][3] = {  
16        { 7, 5, 3 },  
17        { 3, 2, 2 },  
18        { 9, 0, 2 },  
19        { 2, 2, 2 },  
20        { 4, 3, 3 }  
21    };  
22  
23    int avail[3] = { 3, 3, 2 };  
24  
25    int finish[n], ans[n], index = 0;  
26    for (int k = 0; k < n; k++) {  
27        finish[k] = 0;  
28    }  
29  
30    int need[n][totalResources];  
31    for (int i = 0; i < n; i++) {  
32        for (int j = 0; j < totalResources; j++) {  
33            need[i][j] = max[i][j] - alloc[i][j];  
34        }  
35    }  
36  
37    for (int row = 0; row < 5; row++) {  
38        for (int i = 0; i < n; i++) {  
39            if (finish[i] == 0) {  
40                int flag = 0;  
41  
42                for (int r = 0; r < totalResources; r++) {
```



```
for (int r = 0; r < totalResources; r++) {  
    if (need[i][r] > avail[r]) {  
        flag = 1;  
        break;  
    }  
}  
  
if (flag == 0) {  
    ans[index++] = i;  
    for (int y = 0; y < totalResources; y++) {  
        avail[y] += alloc[i][y];  
    }  
    finish[i] = 1;  
    printf("Process P%d is finished.\n", i);  
}  
}  
}  
  
int flag = 1;  
for (int i = 0; i < n; i++) {  
    if (finish[i] == 0) {  
        flag = 0;  
        printf("The system is not in a safe state!\n");  
        break;  
    }  
}  
  
if (flag == 1) {  
    printf("Following is the SAFE Sequence:\n");  
    for (int i = 0; i < n - 1; i++)  
        printf(" P%d -> ", ans[i]);  
    printf(" P%d\n", ans[n - 1]);  
}  
  
return 0;  
}
```

Output:

Run

Enter Input here

If your code takes input, add it in the above box before running.

Output

Status : Successfully executed

Time: 0.0000 secsMemory: 1.628 Mb

Your Output

Process P1 is finished.
Process P3 is finished.
Process P4 is finished.
Process P0 is finished.
Process P2 is finished.
Following is the SAFE Sequence:
P1 -> P3 -> P4 -> P0 -> P2

Conclusion: Hence, we have studied Deadlock avoidance through Bankers Algorithm.