EX.NO: 7 Implement Bankers ALGORITHM for Deadlock Detection
Date :23.09.2024
AIM:

To write a C program to implement the concept of deadlock detection.

ALGORITHM:

1. Get the number of processes and number of resource instances.
2. Get the allocation matrix and Available matrix from the user.
3. Calculate need matrix.
4. Using banker's ALGORITHM allocate resources to processes.
5. Print deadlock occurred or not.
6. Stop the program.

PROGRAM:

```
#include <stdio.h>
#define MAX 100

int max[MAX][MAX], alloc[MAX][MAX], need[MAX][MAX], avail[MAX], n, r;

void input() {
   printf("Enter the number of Processes: ");
   scanf("%d", &n);
   printf("Enter the number of Resource Instances: ");
   scanf("%d", &r);

   printf("Enter the Max Matrix:\n");
   for (int i = 0; i < n; i++)
      for (int j = 0; j < r; j++)
         scanf("%d", &max[i][j]);

   printf("Enter the Allocation Matrix:\n");
   for (int i = 0; i < n; i++)
      for (int j = 0; j < r; j++)
         scanf("%d", &alloc[i][j]);

   printf("Enter the Available Resources:\n");
   for (int j = 0; j < r; j++)
      scanf("%d", &avail[j]);
}

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

```
int isSafe() {
    int finish[MAX] = {0}, work[MAX], count = 0;
    for (int i = 0; i < r; i++)
        work[i] = avail[i];

    while (count < n) {
. int found = 0;
        for (int i = 0; i < n; i++) {
            if (!finish[i]) {
                int j;
                for (j = 0; j < r; j++)
                    if (need[i][j] > work[j]) break;

                if (j == r) {
                    for (j = 0; j < r; j++)
                        work[j] += alloc[i][j];
                    finish[i] = 1;
                    found = 1;
                    count++;
                }
            }
        }
        if (!found) {
            printf("System is in Deadlock. The Deadlocked Processes are:\n");
            for (int i = 0; i < n; i++)
                if (!finish[i])
                    printf("P%d ", i + 1);
            printf("\n");
            return 0; // Deadlock occurred
        }
    }
    printf("No Deadlock Occurred.\n");
    return 1; // No deadlock
}

int main() {
    printf("********** Deadlock Detection Algo **********\n");
    input();
    calculateNeed();
    if (isSafe()) {
        printf("Process\tAllocation\tMax\tAvailable\n");
        for (int i = 0; i < n; i++) {
            printf("P%d\t", i + 1);
            for (int j = 0; j < r; j++)
                printf("%d ", alloc[i][j]);
            printf("\t");
            for (int j = 0; j < r; j++)
                printf("%d ", max[i][j]);
            if (i == 0) {
                printf("\t");
```

```
                for (int j = 0; j < r; j++)
                    printf("%d ", avail[j]);
                }
                printf("\n");
            }
        }
        return 0;
    }
```

OUTPUT:

```
Enter the number of Processes: 5
Enter the number of Resource Instances: 3
Enter the Max Matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the Allocation Matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the Available Resources:
3 3 2

********** Deadlock Detection Algo ************
No Deadlock Occurred.
Process         Allocation      Max     Available
P1      0 1 0   P1: 7 5 3       Available: 3 3 2
P2      2 0 0
P3      3 0 2
P4      2 1 1
P5      0 0 2
```

RESULT:

Thus the banker algorithm was implemented successfully for Deadlock Detection.