**Ex No.: 9 BANKER DEADLOCK AVOIDANCE**

Date : 03.04.2025 **ALGORITHM**

**Aim :**

To find out a safe sequence using Banker’s algorithm for deadlock avoidance.

**Code:**

#include <stdio.h>

#define MAX\_P 10

#define MAX\_R 10

int main() {

int avail[MAX\_R];

int max[MAX\_P][MAX\_R];

int alloc[MAX\_P][MAX\_R];

int need[MAX\_P][MAX\_R];

int done[MAX\_P] = {0};

int work[MAX\_R];

int safe\_seq[MAX\_P];

int p, r, i, j, cnt = 0;

printf("Enter number of processes and resources: ");

scanf("%d %d", &p, &r);

printf("Enter available resources: ");

for (i = 0; i < r; i++) {

scanf("%d", &avail[i]);

work[i] = avail[i];

}

printf("Enter max matrix:\n");

for (i = 0; i < p; i++) {

printf("For process P%d: ", i);

for (j = 0; j < r; j++)

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

}

printf("Enter allocation matrix:\n");

for (i = 0; i < p; i++) {

printf("For process P%d: ", i);

for (j = 0; j < r; j++) {

scanf("%d", &alloc[i][j]);

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

}

}

while (cnt < p) {

int found = 0;

for (i = 0; i < p; i++) {

if (!done[i]) {

int can\_run = 1;

for (j = 0; j < r; j++)

if (need[i][j] > work[j]) can\_run = 0;

if (can\_run) {

for (j = 0; j < r; j++)

work[j] += alloc[i][j];

safe\_seq[cnt++] = i;

done[i] = 1;

found = 1;

}

}

}

if (!found) break;

}

if (cnt == p) {

printf("Safe sequence exists: ");

for (i = 0; i < p; i++)

printf("P%d ", safe\_seq[i]);

} else {

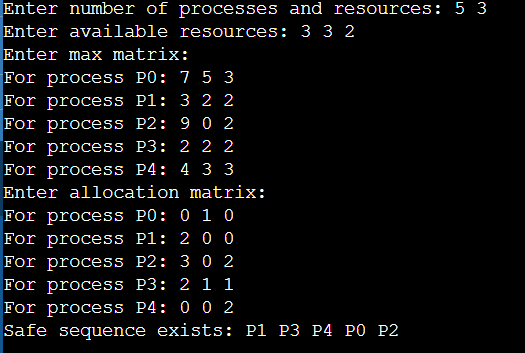
printf("No safe sequence exists (deadlock possible)");

}

return 0;

}

**Output:**

****

**Result:**

Thus the program to find out the safe sequence using Banker’s algorithm for deadlock avoidance has been executed successfully.