**PROGRAM – 7**

**AIM**: To implement Banker’s Algorithm

**INTRODUCTION**: The Banker’s Algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an “s-state” check to test for possible activities, before deciding whether allocation should be allowed to continue. It consists of a safety algorithm which is used for finding out whether or not a system is in safe state. In the following program, we have tried to implement the Banker’s Algorithm in C++ on linux.

**C PROGRAM :**

#include <stdio.h>

int current[5][5], maximum\_claim[5][5], available[5];

int processes,resources;

int need[5][5];

int counter = 0, i, j, k;

int seq[5];

void isSafe(){

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

for (int j = 0; j < resources; ++j){

need[i][j] = maximum\_claim[i][j] - current[i][j];

}

}

int finish[i];

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

finish[i] = 0;

int work[resources];

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

work[i] = available[i];

while(counter< processes){

int p;

int found = 0;

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

if(finish[p] == 0){

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

if (need[p][j]>work[j]){

break;

}

}

if (j == resources){

for ( k = 0; k < resources; ++k){

work[k] += current[p][k];

}

seq[counter] = p;

counter +=1;

finish[p] = 1;

found = 1;

}

}

}

if (found == 0){

printf("\nSystem is Not in Safe State...");

return;

}

}

printf("\nSystem is in Safe State. \nSequence : ");

for (int i = 0; i < processes; ++i){

printf("P%d\t", seq[i] );

}

printf("\n");

}

int main(){

printf("\nEnter number of processes: ");

scanf("%d", &processes);

printf("\nEnter Number of resources: ");

scanf("%d", &resources);

printf("\nEnter Available resources:\n");

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

printf("Resource %d : ",i);

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

}

printf("\nEnter Maximum Resources Table:\n");

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

printf("Process %d : ",i );

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

scanf("%d", &maximum\_claim[i][j]);

}

}

printf("\nEnter Allocated Resources Table:\n");

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

printf("Process %d :", i );

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

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

}

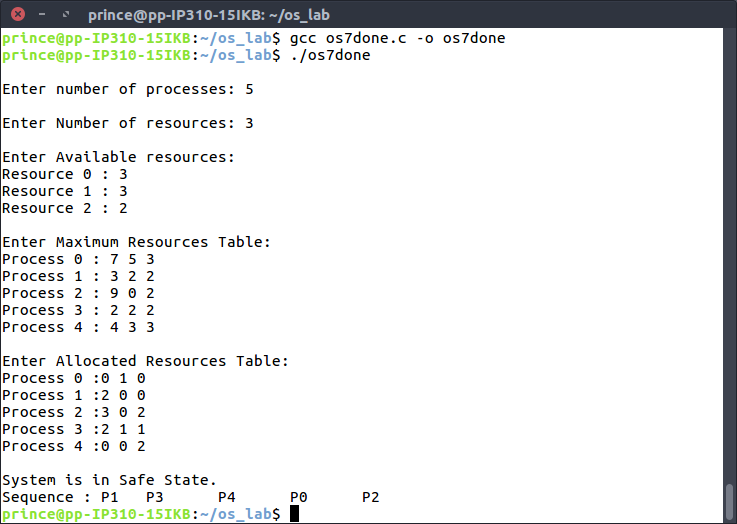
}

isSafe();

return 0;

}

**OUTPUT:**

****

**LEARNING OUTCOMES**

Through the above program, we learnt about the Banker’s Algorithm and how it can be implemented to find out whether a system is in a safe state or not given the various parameters associated with the allocation and need of the resources.