**Experiment-8**

**Banker’s Algorithm**

**Code:**

#include <iostream>

using namespace std;

// Number of processes

const int P = 5;

// Number of resources

const int R = 3;

// Function to find the need of each process

void calculateNeed(int need[P][R], int maxm[P][R], int allot[P][R]){

// calculating Need of each P

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

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

// Need of instance=maximum instance - allocated instance

need[i][j] = maxm[i][j] - allot[i][j];

}

}

}

// Function to find the system is in safe or not

bool inSafe(int processes[], int avail[], int maxm[][R], int allot[][R]){

int need[P][R];

// Function to calculate need matrix

calculateNeed(need, maxm, allot);

// Mark all processes as infinish

bool finish[P] = {0};

// To store safe sequence

int safeSeq[P];

// Make a copy of available resources

int work[R];

for (int i = 0; i < R; i++)

work[i] = avail[i];

// While all processess are not finished or system is not in safe state.

int count = 0;

while (count < P){

// Find a process which is not finish and

// whose needs can be satisfied with current

// work[]resources.

bool found = false;

for (int p = 0; p < P; p++){

// First check if a process is finished,

// if no go for next condition

if (finish[p] == 0){

// Check if for all resources of current p need is lessthan work

int j;

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

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

break;

// if all needs of p were satisfied.

if (j == R){

// Add the allocated resources of current P to the available work resources i.e free the resources

for (int k = 0; k < R; k++)

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

// Add this process to safe sequence.

safeSeq[count++] = p;

// Mark this p as finished

finish[p] = 1;

found = true;

}

}

}

// if we could not find a next process in safe sequence.

if (found == false){

cout << "System is not in safe state";

return false;

}

}

// if system is in safe state then safe sequence will be as below

cout << "System is in safe state.\nSafe"

"sequence is:";

for (int i = 0; i < P; i++)

cout << safeSeq[i] << " ";

return true;

}

int main(){

int processes[] = {0, 1, 2, 3, 4};

// available instances of resources

int avail[] = {3, 3, 2};

// Maximum R that can be allocated to processes

int maxm[][R] = {{7, 5, 3}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2}};

// Resources allocated to processes

int allot[][R] = {{0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2}};

// check system is in safe state or not

inSafe(processes, avail, maxm, allot);

return 0;

}

**Output:**

