**Name: Atharva Salitri Division: CSAI B**

**Roll No.: 37 PRN: 12310120**

**Subject: OS Lab Assignment 5**

**Title: Implementing Banker’s Algorithm**

**1)Safety Algorithm**

**Code:**

#include <stdio.h>

int main() {

int process, resource, i, j, instant, k = 0, count1 = 0, count2 = 0;

printf("Enter number of processes: ");

scanf("%d", &process);

printf("Enter number of Resources: ");

scanf("%d", &resource);

int avail[resource], max[process][resource], allot[process][resource], need[process][resource], completed[process];

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

completed[i] = 0;

printf("Enter available instances of resources: ");

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

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

}

printf("\nEnter max number of instances of resources that each process needs: ");

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

printf("\nFor P[%d]: ", i);

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

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

}

}

printf("\nEnter number of instances already allocated to each process: ");

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

printf("\nFor P[%d]: ", i);

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

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

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

}

}

printf("\nSafe Sequence is: ");

while (count1 != process) {

count2 = count1;

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

if (completed[i] == 0) { // Process not completed

k = 0;

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

if (need[i][j] <= avail[j]) {

k++;

}

}

if (k == resource) { // Process can proceed

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

completed[i] = 1; // Mark process as completed

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

avail[j] += allot[i][j]; // Release allocated resources

}

count1++;

}

}

}

if (count1 == count2) {

printf("\nDeadlock Occurred\n");

break;

}

}

if (count1 == process) {

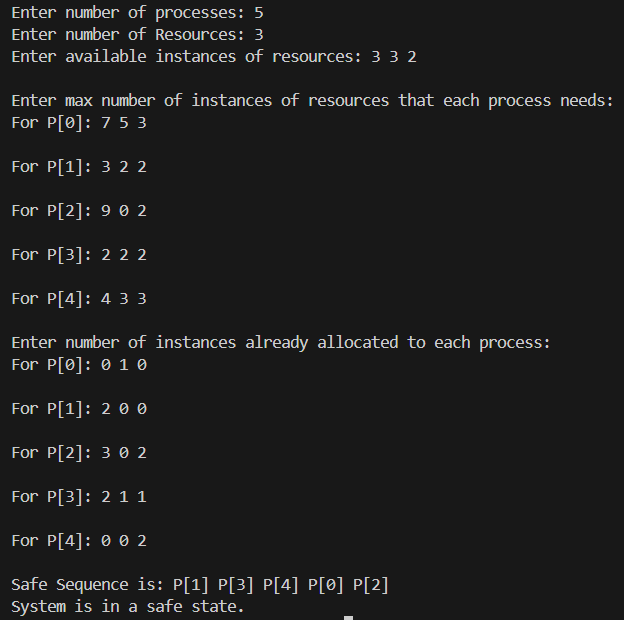
printf("\nSystem is in a safe state.\n");

}

return 0;

}

**Output:**

****

**2)Resource Request Algorithm**

**Code:**

#include <stdio.h>

int main() {

int process, resource, i, j, instant;

int k, count1 = 0, count2 = 0;

printf("Enter number of processes: ");

scanf("%d", &process);

printf("Enter number of Resources: ");

scanf("%d", &resource);

int avail[resource], max[process][resource], allot[process][resource], need[process][resource], completed[process];

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

completed[i] = 0;

printf("Enter available instances of resources: ");

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

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

}

printf("\nEnter max number of instances of resources that each process needs: ");

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

printf("\nFor P[%d]: ", i);

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

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

}

}

printf("\nEnter number of instances already allocated to each process: ");

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

printf("\nFor P[%d]: ", i);

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

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

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

}

}

// Request resources

int request[resource];

int process\_id;

printf("\nEnter the process ID requesting resources (0 to %d): ", process - 1);

scanf("%d", &process\_id);

printf("Enter the request for resources: ");

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

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

}

// Check if request is valid

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

if (request[j] > need[process\_id][j]) {

printf("Error: Process has exceeded its maximum claim.\n");

return 1;

}

if (request[j] > avail[j]) {

printf("Resources are not available right now. Request cannot be granted.\n");

return 1;

}

}

// Pretend to allocate requested resources

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

avail[j] -= request[j];

allot[process\_id][j] += request[j];

need[process\_id][j] -= request[j];

}

// Check for safe state

printf("\nSafe Sequence is: ");

while (count1 != process) {

count2 = count1;

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

if (completed[i] == 0) { // Process not completed

k = 0;

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

if (need[i][j] <= avail[j]) {

k++;

}

}

if (k == resource) { // Process can proceed

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

completed[i] = 1; // Mark process as completed

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

avail[j] += allot[i][j]; // Release allocated resources

}

count1++;

}

}

}

if (count1 == count2) {

printf("\nDeadlock Occurred\n");

return 0; // Exit if deadlock detected

}

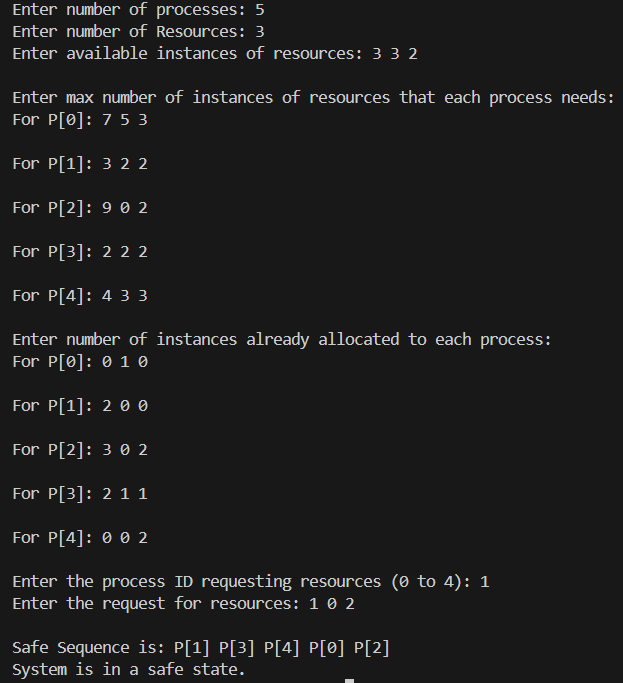
}

printf("\nSystem is in a safe state.\n");

return 0;

}

**Output:**

****