**5. Develop a C program to simulate Bankers Algorithm for DeadLock Avoidance.**

#include <stdio.h>

int i,j;

// Function to check if the system is in a safe state

int isSafe(int processes, int resources, int max[][resources], int allocated[][resources], int available[])

{

int need[processes][resources];

int finish[processes];

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

{

finish[i] = 0;

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

{

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

}

}

int work[resources];

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

{

work[i] = available[i];

}

int safe = 0;

while (1)

{

int found = 0;

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

{

if (!finish[i])

{

int canAllocate = 1;

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

{

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

{

canAllocate = 0;

break;

}

}

if (canAllocate)

{

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

{

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

}

finish[i] = 1;

found = 1;

break;

}

}

}

if (!found)

{

break;

}

}

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

{

if (!finish[i])

{

return 0;

}

}

return 1;

}

int main()

{

int processes, resources;

printf("Enter the number of processes: ");

scanf("%d", &processes);

printf("Enter the number of resources: ");

scanf("%d", &resources);

int max[processes][resources];

int allocated[processes][resources];

int available[resources];

printf("\nEnter the maximum resource matrix:\n");

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

{

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

{

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

}

}

printf("\nEnter the allocated resource matrix:\n");

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

{

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

{

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

}

}

printf("\nEnter the available resources:\n");

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

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

}

if (isSafe(processes, resources, max, allocated, available))

{

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

}

else{

printf("\nThe system is not in a safe state.\n");}

return 0;

}