Experiment – VIII

Aim: WAP to implement Banker’s Algorithm

IDE Used: Dev C++

Code:

#include<iostream>

using namespace std;

const int P = 5;

const int R = 3;

void calculateNeed(int need[P][R], int maxm[P][R],

int allot[P][R])

{

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

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

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

}

bool isSafe(int processes[], int avail[], int maxm[][R],

int allot[][R])

{

int need[P][R];

calculateNeed(need, maxm, allot);

bool finish[P] = {0};

int safeSeq[P];

int work[R];

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

work[i] = avail[i];

int count = 0;

while (count < P)

{

bool found = false;

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

{

if (finish[p] == 0)

{

int j;

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

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

break;

if (j == R)

{

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

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

safeSeq[count++] = p;

finish[p] = 1;

found = true;

}

}

}

if (found == false)

{

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

return false;

}

}

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};

int avail[] = {5, 5, 3};

int maxm[][R] = {{7, 5, 3},

{3, 2, 2},

{1, 6, 2},

{2, 2, 2},

{4, 4, 3}};

int allot[][R] = {{0, 1, 0},

{2, 0, 0},

{3, 1, 2},

{2, 1, 1},

{0, 0, 2}};

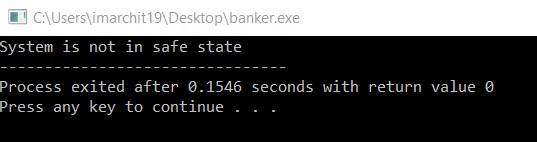
isSafe(processes, avail, maxm, allot);

return 0;

}

OUTPUT

1. System not safe
   1. Input Avail: {8, 2, 8}
   2. Output:

’

1. System is safe
   1. Input Avail: {5, 5, 3}
   2. Output:

