Practical 6

AIM: Write a program that implements RR scheduling algorithm.

CODE:

public class GFGBankers {

int n = 5;

int m = 3;

int need[][] = new int[n][m];

int[][] max;

int[][] alloc;

int[] avail;

int safeSequence[] = new int[n];

void initializeValues() {

alloc = new int[][] { { 0, 1, 0 }, { 2, 0, 0 }, { 3, 0, 2 }, { 2, 1, 1 }, { 0, 0, 2 } };

max = new int[][] { { 7, 5, 3 }, { 3, 2, 2 }, { 9, 0, 2 }, { 2, 2, 2 }, { 4, 3, 3 } };

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

}

void isSafe() {

int count = 0;

boolean visited[] = new boolean[n];

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

visited[i] = false;

}

int work[] = new int[m];

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

work[i] = avail[i];

}

while (count < n) {

boolean flag = false;

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

if (visited[i] == false) {

int j;

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

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

break;

}

if (j == m) {

safeSequence[count++] = i;

visited[i] = true;

flag = true;

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

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

}

}

}

}

if (flag == false) {

break;

}

}

if (count < n) {

System.out.println("The System is Unsafe!");

} else {

System.out.println("Following is the safe sequence");

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

System.out.print("P" + safeSequence[i]);

if (i != n - 1)

System.out.print("->");

}

}

}

void calculateNeed() {

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

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

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

}

}

}

public static void main(String[] args) {

int i, j, k;

GFGBankers gfg = new GFGBankers();

gfg.initializeValues();

gfg.isSafe();

}

}

OUTPUT:

Following is the safe sequence

P0->P1->P2->P3->P4