#include <stdio.h>

#include <unistd.h>

#define BUFFER\_SIZE 5

#define TOTAL\_ITEMS 10

Int buffer[BUFFER\_SIZE];

Int count = 0;

Int produced = 0, consumed = 0;

Void producer() {

If (count < BUFFER\_SIZE && produced < TOTAL\_ITEMS) {

Buffer[count++] = ++produced;

Printf(“Produced: %d\n”, produced);

} else if (count == BUFFER\_SIZE) {

Printf(“Buffer full, producer waiting…\n”);

}

}

Void consumer() {

If (count > 0 && consumed < TOTAL\_ITEMS) {

Int item = buffer[--count];

Consumed++;

Printf(“Consumed: %d\n”, item);

} else if (count == 0) {

Printf(“Buffer empty, consumer waiting…\n”);

}

}

Int main() {

Printf(“Starting Producer-Consumer Simulation…\n”);

While (produced < TOTAL\_ITEMS || consumed < TOTAL\_ITEMS) {

If (produced < TOTAL\_ITEMS) {

Producer();

}

If (consumed < produced) {

Consumer();

}

Sleep(1); // Simulate time delay

}

Printf(“Simulation completed!\n”);

Return 0;

}

#include <stdio.h>

#define MAX\_PROCESSES 10

#define MAX\_RESOURCES 10

Int main() {

Int n, m, I, j, k;

Printf(“Enter the number of processes (max %d): “, MAX\_PROCESSES);

Scanf(“%d”, &n);

Printf(“Enter the number of resources (max %d): “, MAX\_RESOURCES);

Scanf(“%d”, &m);

Int alloc[MAX\_PROCESSES][MAX\_RESOURCES];

Int max[MAX\_PROCESSES][MAX\_RESOURCES];

Int avail[MAX\_RESOURCES];

Int finish[MAX\_PROCESSES] = {0};

Int safe\_sequence[MAX\_PROCESSES];

Int need[MAX\_PROCESSES][MAX\_RESOURCES];

Printf(“\nEnter the Allocation matrix (%d x %d):\n”, n, m);

For (I = 0; I < n; i++) {

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

Scanf(“%d”, &alloc[i][j]);

}

}

Printf(“\nEnter the Maximum matrix (%d x %d):\n”, n, m);

For (I = 0; I < n; i++) {

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

Scanf(“%d”, &max[i][j]);

}

}

Printf(“\nEnter the Available resources (%d):\n”, m);

For (I = 0; I < m; i++) {

Scanf(“%d”, &avail[i]);

}

For (I = 0; I < n; i++) {

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

Need[i][j] = max[i][j] – alloc[i][j];

}

}

Int count = 0;

While (count < n) {

Int found = 0;

For (I = 0; I < n; i++) {

If (finish[i] == 0) {

Int flag = 1;

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

If (need[i][j] > avail[j]) {

Flag = 0;

Break;

}

}

If (flag) {

For (k = 0; k < m; k++) {

Avail[k] += alloc[i][k];

}

Safe\_sequence[count++] = I;

Finish[i] = 1;

Found = 1;

}

}

}

If (!found) {

Printf(“The system is in an unsafe state.\n”);

Return 1;

}

}

Printf(“The system is in a safe state.\nSafe sequence: “);

For (I = 0; I < n; i++) {

Printf(“%d “, safe\_sequence[i]);

}

Printf(“\n”);

Return 0;

}