1: #include <pthread.h>

2: #include <semaphore.h>

3: #include <stdlib.h>

4: #include <stdio.h>

5: #include <stdint.h> // For intptr\_t

6: #include <time.h> // For time function

8: #define MaxItems 5 // Maximum items a producer can produce or a consumer can consume 9: #define BufferSize 5 // Size of the buffer

11: sem\_t empty;

12: sem\_t full;

13: int in = 0;

14: int out = 0;

15: int buffer[BufferSize]; pthread\_mutex\_t mutex;

18: void \*producer(void \*pno) 19: {

20: int item;

21: for (int i = 0; i < MaxItems; i++) {

22: item = rand(); // Produce a random item 23: sem\_wait(&empty);

24: pthread\_mutex\_lock(&mutex); 26: buffer[in] = item;

27: printf("Producer %d: Insert Item %d at %d\n", \*((int \*)pno), buffer[in], in); 28: in = (in + 1) % BufferSize;

30: pthread\_mutex\_unlock(&mutex);

31: sem\_post(&full);

32: }

33: return NULL; // Added return 34: }

36: void \*consumer(void \*cno) 37: {

38: for (int i = 0; i < MaxItems; i++) { 39: sem\_wait(&full);

40: pthread\_mutex\_lock(&mutex); 42: int item = buffer[out];

43: printf("Consumer %d: Remove Item %d from %d\n", \*((int \*)cno), item, out); 44: out = (out + 1) % BufferSize;

46: pthread\_mutex\_unlock(&mutex);

47: sem\_post(&empty);

48: }

49: return NULL; // Added return 50: }

52: int main()

53: {

54: pthread\_t pro[5], con[5];

55: pthread\_mutex\_init(&mutex, NULL);

56: sem\_init(&empty, 0, BufferSize); // Initialize empty semaphore to BufferSize 57: sem\_init(&full, 0, 0); // Initialize full semaphore to 0

59: int a[5] = {1, 2, 3, 4, 5}; // Just used for numbering the producer and consumer 60: srand((unsigned int)time(NULL)); // Seed the random number generator

62: for (int i = 0; i < 5; i++) {

63: pthread\_create(&pro[i], NULL, producer, (void \*)&a[i]);

64: }

65: for (int i = 0; i < 5; i++) {

66: pthread\_create(&con[i], NULL, consumer, (void \*)&a[i]); 67: }

69: for (int i = 0; i < 5; i++) {

70: pthread\_join(pro[i], NULL);

71: }

72: for (int i = 0; i < 5; i++) {

73: pthread\_join(con[i], NULL);

74: }

76: pthread\_mutex\_destroy(&mutex);

77: sem\_destroy(&empty);

78: sem\_destroy(&full);

80: return 0;

81: }

