

```

The file "/home/minhduc/Desktop/Lab5/pro_con" changed on disk.
1 // full = 0;
2 // empty variable will store the number of empty slots in the buffer.
3 int empty = 10, data = 0;
4 // A function that will resemble producers' production of data
5 void producer()
6 {
7     // decrementing the value of mutex
8     --mutex;
9     // Increase the number of full slots
10    ++full;
11    // decrementing the number of slots available
12    --empty;
13    // incrementing data which means that the data is produced.
14    data++;
15    printf("\nproducer produces item number: %d\n", data);
16    // incrementing the value of mutex
17    ++mutex;
18 }
19 // A function that will resemble the consumer's consumption of data
20 void consumer()
21 {
22     // decrementing the value of mutex
23     --mutex;
24     // Decrease the number of full slots
25     --full;
26     // incrementing the number of slots available
27     ++empty;
28     printf("\nconsumer consumes item number: %d\n", data);
29     // since data is consumed, let us decrease the value of data
30     data--;
31     // incrementing the value of mutex
32     ++mutex;
33 }
34
35 int main()
36 {
37     producer();
38     consumer();
39     return 0;
40 }

```

```

minhduc@minhduc-VirtualBox: ~/Desktop/Lab5
Producer produces item number: 10
Enter your choice: 1
The Buffer is full. New data cannot be produced!
Enter your choice: 2
Consumer consumes item number: 10.
Enter your choice: 2
Consumer consumes item number: 9.
Enter your choice: 2
Consumer consumes item number: 8.
Enter your choice: 2
Consumer consumes item number: 7.
Enter your choice: 2
Consumer consumes item number: 6.
Enter your choice: 2
Consumer consumes item number: 5.
Enter your choice: 2
Consumer consumes item number: 4.
Enter your choice: 2
Consumer consumes item number: 3.
Enter your choice: 2
Consumer consumes item number: 2.
Enter your choice: 2
Consumer consumes item number: 1.
Enter your choice: 2
The Buffer is empty! New data cannot be consumed!
Enter your choice: 1

```

Producer and consumer test

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4
5 #define BUFFER_SIZE 5
6 #define MAX_ITEMS 5
7
8 int buffer[BUFFER_SIZE];
9 int in = 0;
10 int out = 0;
11 int produced_count = 0;
12 int consumed_count = 0;
13
14 pthread_mutex_t mutex;
15 pthread_cond_t full;
16 pthread_cond_t empty;
17
18 void* producer(void* arg) {
19     int item = 1;
20     while (produced_count < MAX_ITEMS) {
21         pthread_mutex_lock(&mutex);
22         while (((in + 1) % BUFFER_SIZE) == out) {
23             pthread_cond_wait(&empty, &mutex);
24         }
25         buffer[in] = item;
26         printf("Produced: %d\n", item);
27         item++;
28         in = (in + 1) % BUFFER_SIZE;
29         produced_count++;
30         pthread_cond_signal(&full);
31         pthread_mutex_unlock(&mutex);
32     }
33     pthread_exit(NULL);
34 }
35
36 void* consumer(void* arg) {
37     while (consumed_count < MAX_ITEMS) {
38         pthread_mutex_lock(&mutex);
39         while (out == in) {
40             pthread_cond_wait(&empty, &mutex);
41         }
42         item = buffer[out];
43         printf("Consumed: %d\n", item);
44         out = (out + 1) % BUFFER_SIZE;
45         consumed_count++;
46         pthread_mutex_unlock(&mutex);
47     }
48     pthread_exit(NULL);
49 }
50
51 int main() {
52     pthread_t producer_thread;
53     pthread_t consumer_thread;
54     pthread_create(&producer_thread, NULL, producer, NULL);
55     pthread_create(&consumer_thread, NULL, consumer, NULL);
56     pthread_join(producer_thread, NULL);
57     pthread_join(consumer_thread, NULL);
58     return 0;
59 }

```

```

minhduc@minhduc-VirtualBox: ~/Desktop/Lab5
minhduc@minhduc-VirtualBox: ~/Desktop/Lab5$ gcc pro_con_thread.c -o pro_con_thread
minhduc@minhduc-VirtualBox: ~/Desktop/Lab5$ ./pro_con_thread
Produced: 1
Produced: 2
Produced: 3
Produced: 4
Consumed: 1
Consumed: 2
Consumed: 3
Consumed: 4
Produced: 5
Consumed: 5
minhduc@minhduc-VirtualBox: ~/Desktop/Lab5$

```

Producer and consumer using threads

```

semaphores.c
33 sem_post(&mutex);
34 sem_post(&full);
35 }
36
37 pthread_exit(NULL);
38 }
39
40 void* consumer(void* arg) {
41     while (consumed_count < MAX_ITEMS) {
42         sem_wait(&full);
43         sem_wait(&mutex);
44
45         int item = buffer[out];
46         printf("consumed: %d\n", item);
47         out = (out + 1) % BUFFER_SIZE;
48         consumed_count++;
49         sem_post(&mutex);
50         sem_post(&empty);
51     }
52     pthread_exit(NULL);
53 }
54
55 int main() {
56     pthread_t producerThread, consumerThread;
57
58     sem_init(&mutex, 0, 1);
59     sem_init(&full, 0, 0);
60     sem_init(&empty, 0, BUFFER_SIZE);
61
62     pthread_create(&producerThread, NULL, producer, NULL);
63     pthread_create(&consumerThread, NULL, consumer, NULL);
64
65     pthread_join(producerThread, NULL);
66     pthread_join(consumerThread, NULL);
67
68     sem_destroy(&mutex);
69     sem_destroy(&full);
70     sem_destroy(&empty);
71
72     return 0;
73 }
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

minhduc@minhduc-VirtualBox: ~/Desktop/Lab5
minhduc@minhduc-VirtualBox:~/Desktop/Lab5$ gcc semaphores.c -o semaphores
minhduc@minhduc-VirtualBox:~/Desktop/Lab5$ ./semaphores
Produced: 1
Produced: 2
Produced: 3
Produced: 4
Produced: 5
Consumed: 1
Consumed: 2
Consumed: 3
Consumed: 4
Consumed: 5
Produced: 6
Produced: 7
Produced: 8
Produced: 9
Produced: 10
Consumed: 6
Consumed: 7
Consumed: 8
Consumed: 9
Consumed: 10
Produced: 11
Produced: 12
Produced: 13
Produced: 14
Produced: 15
Consumed: 11
Consumed: 12
Consumed: 13
Consumed: 14
Consumed: 15
Produced: 16
Produced: 17
Produced: 18
Produced: 19
Produced: 20
Consumed: 16
Consumed: 17
Consumed: 18
Consumed: 19
Consumed: 20
minhduc@minhduc-VirtualBox:~/Desktop/Lab5$

```

Producer and consumer using semaphores

```

problem4.2.c
31 void* consumer(void* arg) {
32     int item;
33
34     for (int i = 0; i < 10; i++) {
35         sem_wait(&full);
36         sem_wait(&mutex);
37
38         item = buffer[out];
39         printf("consumer consumed: %d\n", item);
40         out = (out + 1) % BUFFER_SIZE;
41         sem_post(&mutex);
42         sem_post(&empty);
43     }
44     pthread_exit(NULL);
45 }
46
47 int main() {
48     pthread_t producer_thread, consumer1_thread, consumer2_thread;
49
50     // Initialize semaphores
51     sem_init(&mutex, 0, 1);
52     sem_init(&empty, 0, BUFFER_SIZE);
53     sem_init(&full, 0, 0);
54
55     // Create producer and consumer threads
56     pthread_create(&producer_thread, NULL, producer, NULL);
57     pthread_create(&consumer1_thread, NULL, consumer, NULL);
58     pthread_create(&consumer2_thread, NULL, consumer, NULL);
59
60     // Wait for threads to finish
61     pthread_join(producer_thread, NULL);
62     pthread_join(consumer1_thread, NULL);
63     pthread_join(consumer2_thread, NULL);
64
65     // Clean up semaphores
66     sem_destroy(&mutex);
67     sem_destroy(&empty);
68     sem_destroy(&full);
69
70     return 0;
71 }
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

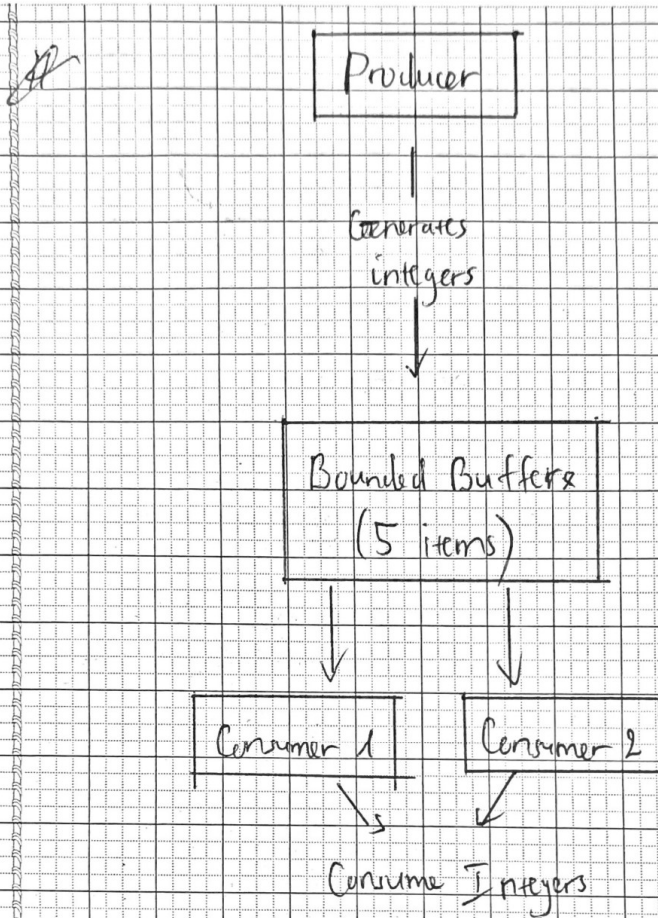
```

minhduc@minhduc-VirtualBox: ~/Desktop/Lab5
minhduc@minhduc-VirtualBox:~/Desktop/Lab5$ gcc problem4.2.c -o problem4.2
minhduc@minhduc-VirtualBox:~/Desktop/Lab5$ ./problem4.2
Producer produced: 83
Producer produced: 86
Producer produced: 77
Producer produced: 15
Producer produced: 93
Consumer consumed: 83
Consumer consumed: 86
Consumer consumed: 77
Consumer consumed: 15
Consumer consumed: 93
Producer produced: 35
Producer produced: 86
Producer produced: 92
Producer produced: 49
Consumer consumed: 35
Consumer consumed: 86
Consumer consumed: 21
Producer produced: 92
Consumer consumed: 92
Consumer consumed: 49
Consumer consumed: 21

```

Problem 4.2

Diagram



- The ~~pro~~ "Producer" generates integers and places them in the "Bounded Buffer"
- The "Bounded Buffer" is a shared resource with a size of 5 items. The producer inserts ~~the~~ items and the consumer retrieve items
- "Consumer 1" and "Consumer 2" are the consumers who consume integers from the "Bounded Buffer"