

```
// C program for the above approach
#include <stdio.h>
#include <stdlib.h>
// Initialize a mutex to 1
int mutex = 1;
// Number of full slots as 0
int full = 0;
// Number of empty slots as size
// of buffer
int empty = 10, x = 0;
// Function to produce an item and
// add it to the buffer
void producer()
{
    // Decrease mutex value by 1 --mutex;
    // Increase the number of full
    // slots by 1
    ++full;
    // Decrease the number of empty
    // slots by 1
    --empty;

    // Item produced
    x++;
    printf("\nProducer produces"
        "item %d",
        x);

    // Increase mutex value by 1
    ++mutex;
}

// Function to consume an item and
// remove it from buffer
void consumer()
{
    // Decrease mutex value by 1
    --mutex;

    // Decrease the number of full
    // slots by 1
    --full;

    // Increase the number of empty
```

```

// slots by 1
++empty;
printf("\nConsumer consumes "
    "item %d",
    x);
x--;
// Increase mutex value by 1
++mutex;
}

```

```

// Driver Code
int main()
{
    int n, i;
    printf("\n1. Press 1 for Producer"
        "\n2. Press 2 for Consumer"
        "\n3. Press 3 for Exit");

```

```

// Using '#pragma omp parallel for'
// can give wrong value due to
// synchronization issues.

```

```

3
// 'critical' specifies that code is
// executed by only one thread at a
// time i.e., only one thread enters
// the critical section at a given time
#pragma omp critical

```

```

for (i = 1; i > 0; i++) {

    printf("\nEnter your choice:");
    scanf("%d", &n);

```

```

    // Switch Cases
    switch (n) {
        case 1:
            // If mutex is 1 and empty
            // is non-zero, then it is
            // possible to produce
            if ((mutex == 1)
                && (empty != 0)) {
                producer();
            }
    }
}

```

```
// Otherwise, print buffer
// is full
else {
    printf("Buffer is full!");
}
break;
```

case 2:

```
// If mutex is 1 and full
// is non-zero, then it is
// possible to consume
if ((mutex == 1)
    && (full != 0)) {
    consumer();
}

// Otherwise, print Buffer
// is empty
else {
    printf("Buffer is empty!");
}
break;
// Exit Condition
case 3:
    exit(0);
    break;
}
```