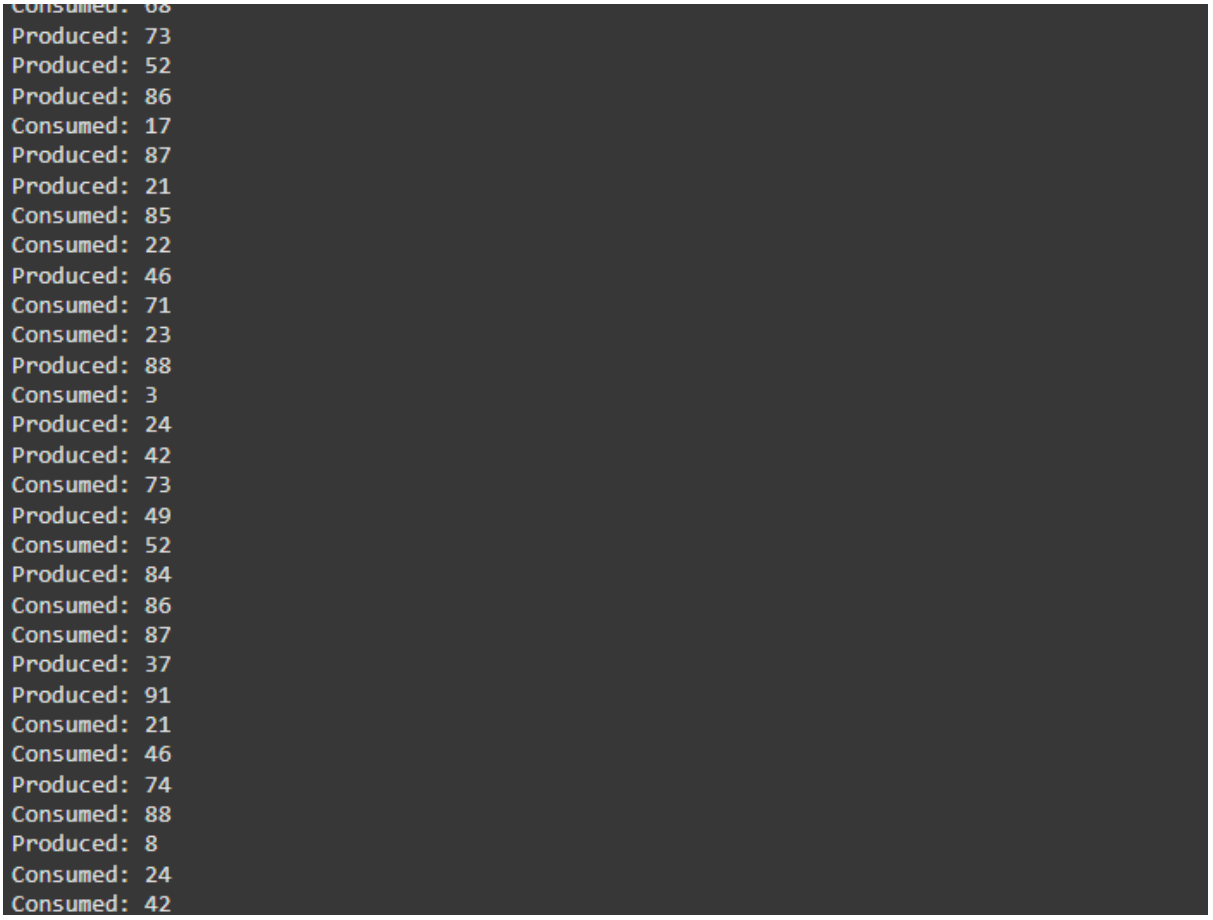
Lab4

Q1.

```
1. Produce   2. Consume     3. Exit
Enter your choice: 2

Buffer is Empty
1. Produce   2. Consume     3. Exit
Enter your choice: 1

Enter the value: 2

1. Produce   2. Consume     3. Exit
Enter your choice: 3

Exiting...

=== Code Execution Successful ===
```

Q2.

```
Buffer is full. Producer is waiting...
Consumed: 11
Produced: 35
Buffer is full. Producer is waiting...
Consumed: 47
Produced: 11
Consumed: 21
Consumed: 90
Consumed: 13
Produced: 25
Consumed: 11
Consumed: 38
Produced: 42
Consumed: 71
Produced: 37
Produced: 85
Consumed: 82
Produced: 68
Produced: 17
Consumed: 53
Consumed: 35
Consumed: 11
Produced: 85
Produced: 22
Produced: 71
Consumed: 25
Produced: 23
```



Q3.

Using a **stack** instead of a **queue-based buffer (array)** in the producer-consumer problem changes the order of consumption from **FIFO (First In, First Out)** to **LIFO (Last In, First Out)**, meaning the most recently produced item is consumed first. This can be useful in scenarios where the latest data is more relevant, but it may lead to older items being delayed or never processed if production is continuous. Additionally, a stack-based approach might cause starvation issues in time-sensitive applications, whereas a queue ensures fair processing. However, both require proper synchronization mechanisms to handle concurrent access.