

## LAB 04

1) Implement the above code and paste the screen shot of the output.

### CODE

```
#include <stdio.h>

int main() {
    int buffer[10], bufsize, in, out, produce, consume, choice = 0;
    in = 0;
    out = 0;
    bufsize = 10;

    while (choice != 3) {
        printf("\n1. Produce \t 2. Consume \t 3. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                if ((in + 1) % bufsize == out)
                    printf("\nBuffer is Full");
                else {
                    printf("\nEnter the value: ");
                    scanf("%d", &produce);
                    buffer[in] = produce;
                    in = (in + 1) % bufsize;
                }
                break;

            case 2:
                if (in == out)
                    printf("\nBuffer is Empty");
                else {
                    consume = buffer[out];
                    printf("\nThe consumed value is %d", consume);
                    out = (out + 1) % bufsize;
                }
                break;
        }
    }
}
```

## OUTPUT

```

1  #include <stdio.h>
2
3  int main() {
4      C:\Users\admin\Downloads\DM lab 04.exe
5
6      1. Produce      2. Consume      3. Exit
7      Enter your choice: 2
8
9      Buffer is Empty
10     1. Produce      2. Consume      3. Exit
11     Enter your choice: 1
12
13     Enter the value: 5
14
15     1. Produce      2. Consume      3. Exit
16     Enter your choice: 2
17
18     The consumed value is 5
19     1. Produce      2. Consume      3. Exit
20     Enter your choice: 1
21
22     Enter the value: 54
23
24     1. Produce      2. Consume      3. Exit
25     Enter your choice: 1
26
27     Enter the value: 2
28
29     1. Produce      2. Consume      3. Exit
30     Enter your choice: 2
31
32     The consumed value is 54
33     1. Produce      2. Consume      3. Exit
34     Enter your choice: 50
35
36     Warnings: 0
37     - Output Filename: C:\Users\admin\Downloads\DM lab 04.exe
38     - Output Size: 108,541,665,748

```

```

6     Enter your choice: 1
7     Enter the value: 2
8
9     1. Produce      2. Consume      3. Exit
10    Enter your choice: 2
11
12    The consumed value is 54
13    1. Produce      2. Consume      3. Exit
14    Enter your choice: 50
15
16    1. Produce      2. Consume      3. Exit
17    Enter your choice: 1
18
19    Enter the value: 20
20
21    1. Produce      2. Consume      3. Exit
22    Enter your choice: 2
23
24    The consumed value is 2
25    1. Produce      2. Consume      3. Exit
26    Enter your choice: 3
27
28    -----
29    Process exited after 59.81 seconds with return value 0
30    Press any key to continue . . .

```

**2) Solve the producer-consumer problem using linked list. Note: Keep the buffer size to 10 places.**

**CODE**

```
#include <stdio.h>
#define BUFFER_SIZE 10

typedef struct Node {
    int data;
    struct Node* next;
} Node;

Node* head = NULL;
Node* tail = NULL;
int count = 0;

pthread_mutex_t mutex;
sem_t empty, full;

void insert(int item) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = item;
    newNode->next = NULL;

    if (tail == NULL) {
        head = tail = newNode;
    } else {
        tail->next = newNode;
        tail = newNode;
    }
    count++;
}

int remove_item() {
    if (head == NULL) return -1;

    Node* temp = head;
    int item = temp->data;
    head = head->next;
```

```
    if (head == NULL) tail = NULL;

    free(temp);
    count--;
    return item;
}

void* producer(void* arg) {
    int item;
    while (1) {
        item = rand() % 100;
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);

        insert(item);
        printf("Produced: %d\n", item);

        pthread_mutex_unlock(&mutex);
        sem_post(&full);
        sleep(1);
    }
}

void* consumer(void* arg) {
    int item;
    while (1) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);

        item = remove_item();
        printf("Consumed: %d\n", item);

        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
        sleep(1);
    }
}

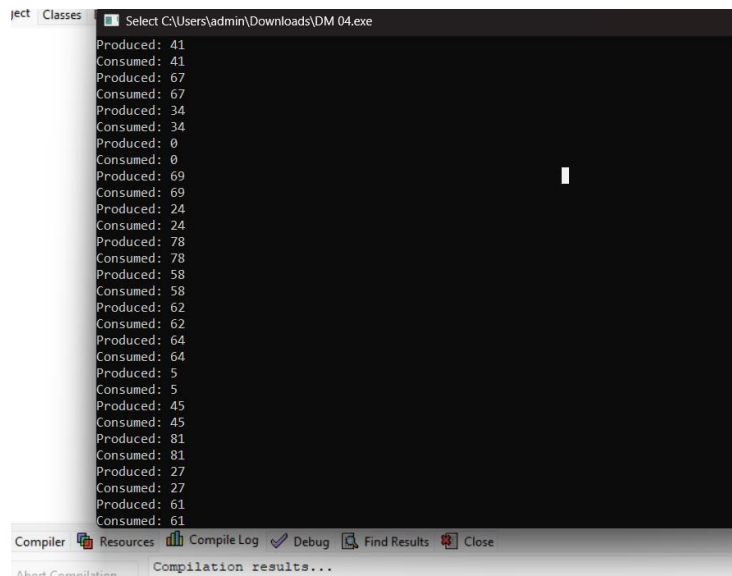
int main() {
    pthread_t prod, cons;

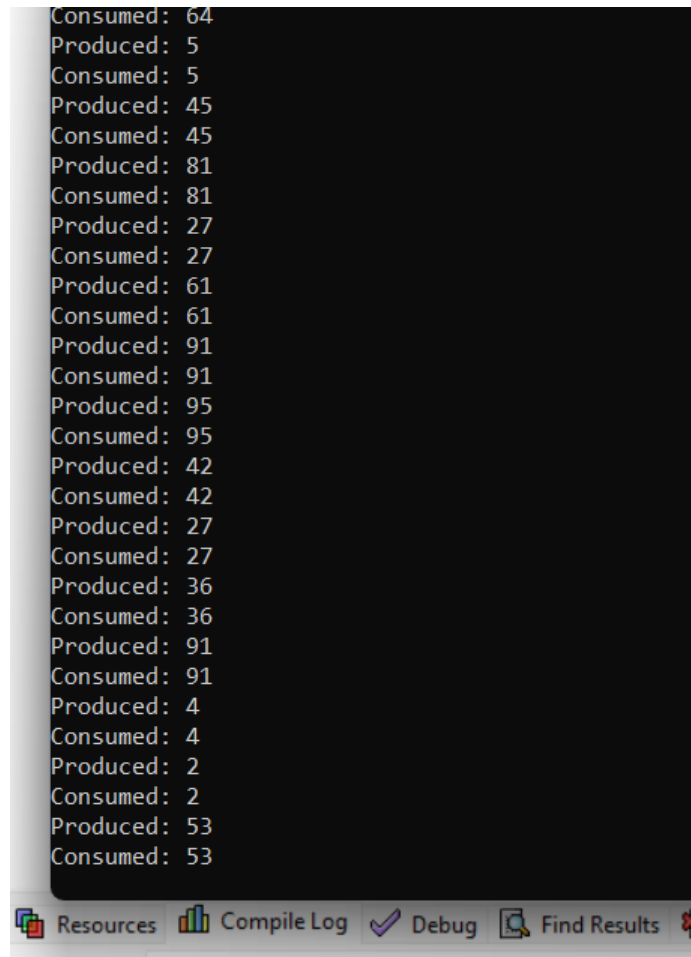
    pthread_mutex_init(&mutex, NULL);
    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);

    pthread_create(&prod, NULL, producer, NULL);
    pthread_create(&cons, NULL, consumer, NULL);
```

```
pthread_join(prod, NULL);  
pthread_join(cons, NULL);  
  
pthread_mutex_destroy(&mutex);  
sem_destroy(&empty);  
sem_destroy(&full);  
return 0;  
}
```

### Output





```
Consumed: 64
Produced: 5
Consumed: 5
Produced: 45
Consumed: 45
Produced: 81
Consumed: 81
Produced: 27
Consumed: 27
Produced: 61
Consumed: 61
Produced: 91
Consumed: 91
Produced: 95
Consumed: 95
Produced: 42
Consumed: 42
Produced: 27
Consumed: 27
Produced: 36
Consumed: 36
Produced: 91
Consumed: 91
Produced: 4
Consumed: 4
Produced: 2
Consumed: 2
Produced: 53
Consumed: 53
```

**3) In producer-consumer problem what difference will it make if we utilize stack for the buffer rather than an array?**

Using a stack instead of a queue in the producer-consumer problem fundamentally changes the processing order from FIFO to LIFO, which may not be suitable for many traditional producer-consumer use cases.