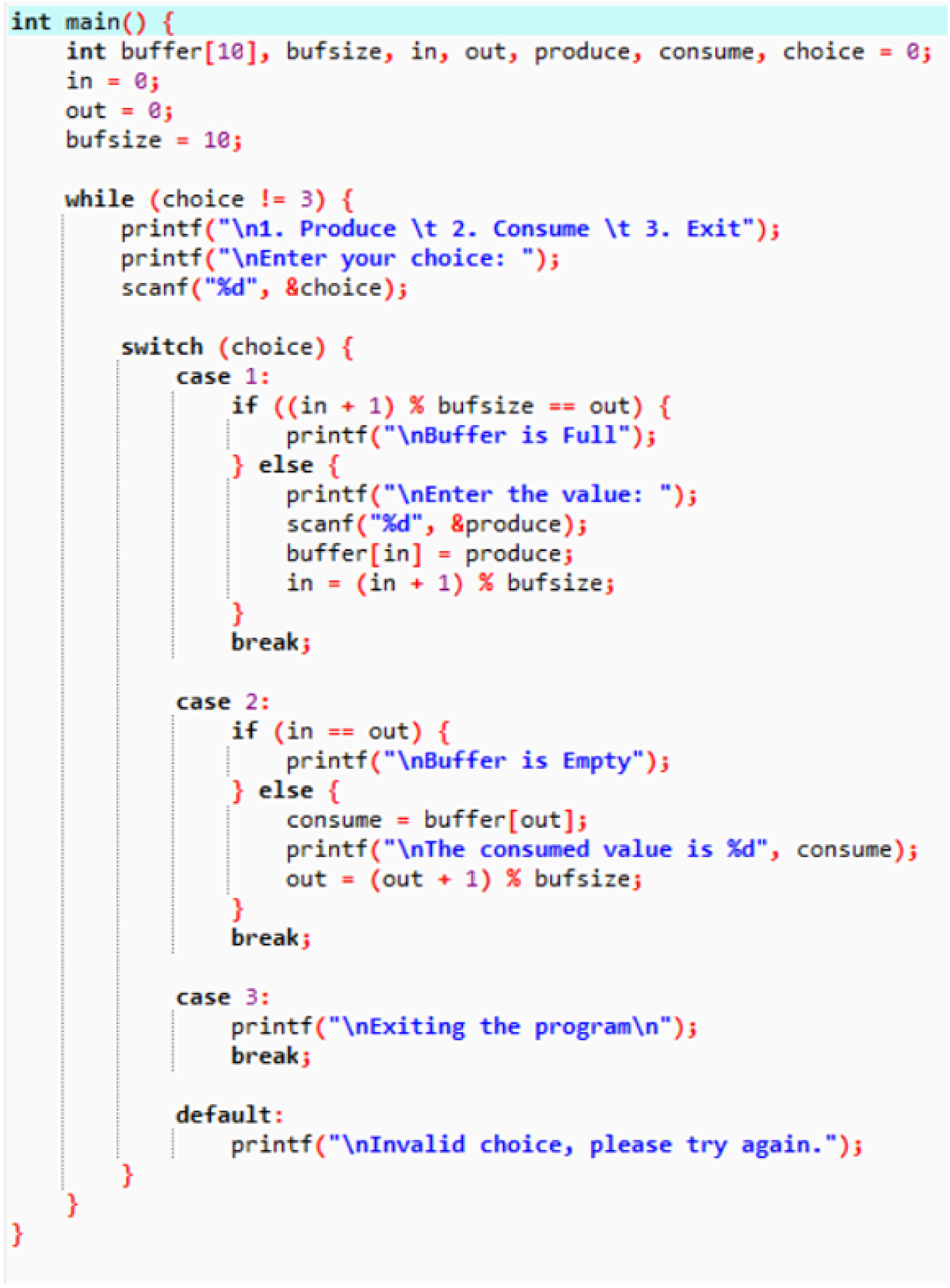
ASHNA SHAIKH [DT-22019]

OPERATING SYSTEM LAB [CT-353]

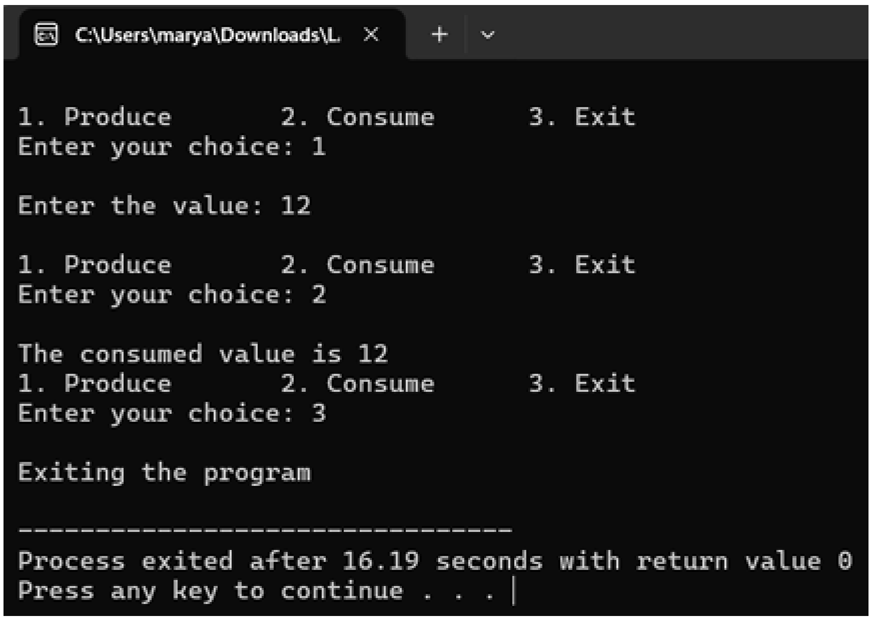
—---

LAB 04

1) Implement the above code and paste the screenshot of the output.

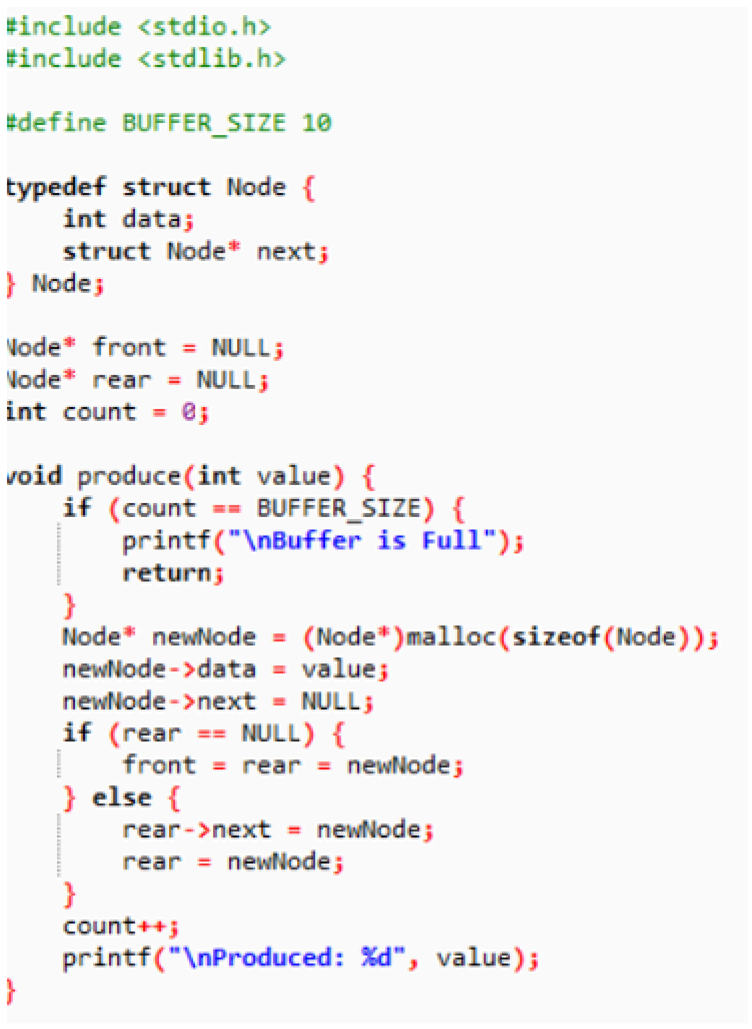


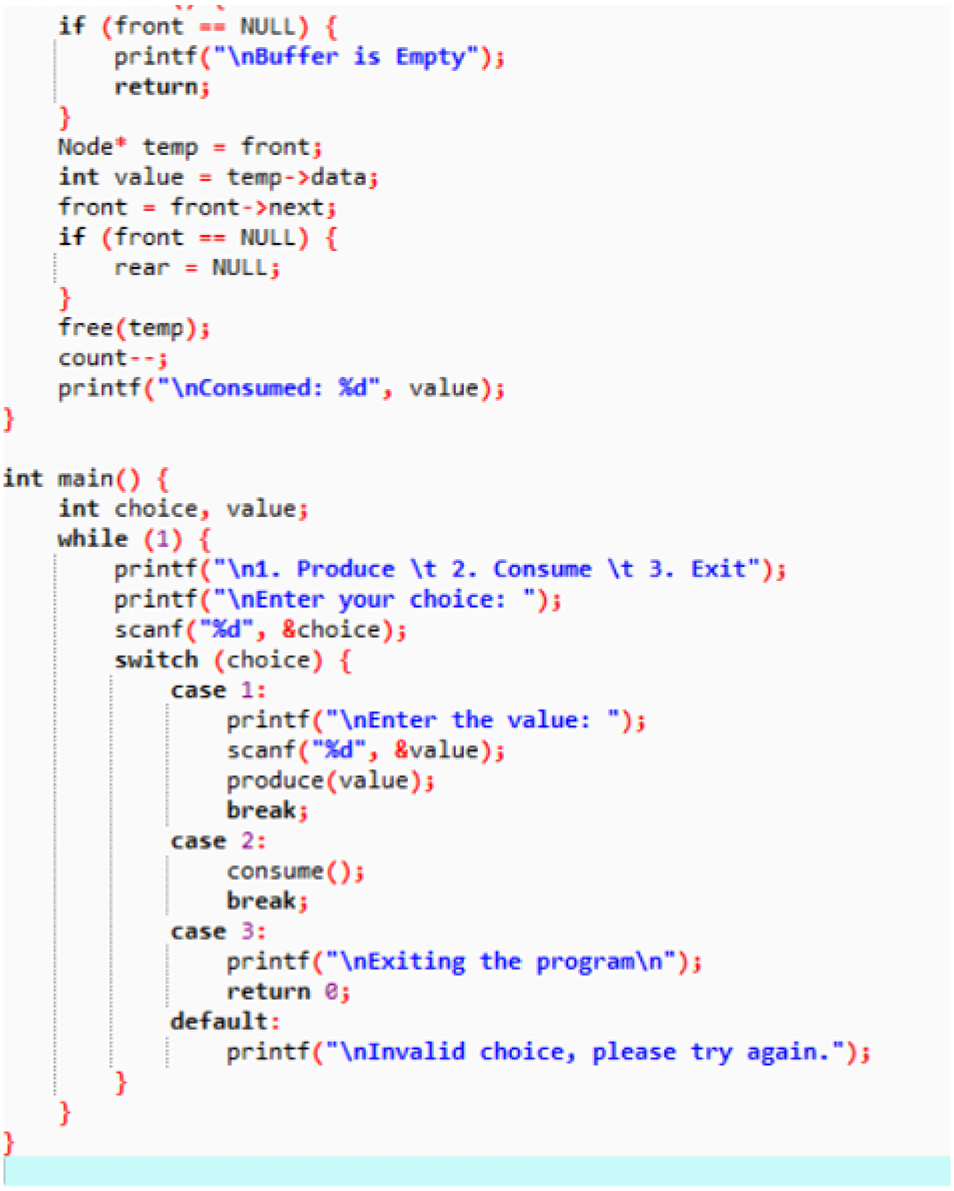
```c
int main() {
    int buffer[10], bufsize, in, out, produce, consume, choice = 0;
    in = 0;
    out = 0;
    bufsize = 10;

    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;

            case 3:
                printf("\nExiting the program\n");
                break;

            default:
                printf("\nInvalid choice, please try again.");
        }
    }
}
```



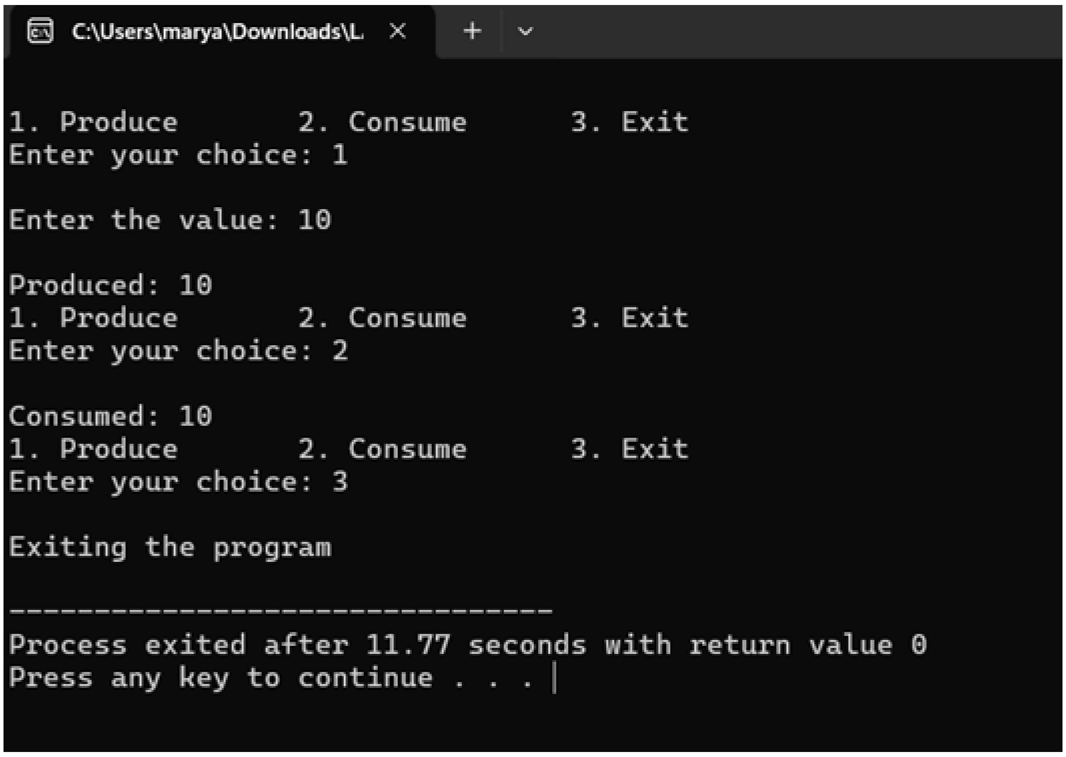
```
1. Produce       2. Consume      3. Exit
Enter your choice: 1

Enter the value: 12

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

The consumed value is 12
1. Produce       2. Consume      3. Exit
Enter your choice: 3

Exiting the program

--------------------------------
Process exited after 16.19 seconds with return value 0
Press any key to continue . . .
```

2) Solve the producer-consumer problem using linked list. (You can perform this task using any programming language) Note: Keep the buffer size to 10 places.


```c
#include <stdio.h>
#include <stdlib.h>

#define BUFFER_SIZE 10

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

Node* front = NULL;
Node* rear = NULL;
int count = 0;

void produce(int value) {
    if (count == BUFFER_SIZE) {
        printf("\nBuffer is Full");
        return;
    }
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = NULL;
    if (rear == NULL) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
    count++;
    printf("\nProduced: %d", value);
}
```



```c
    if (front == NULL) {
        printf("\nBuffer is Empty");
        return;
    }
    Node* temp = front;
    int value = temp->data;
    front = front->next;
    if (front == NULL) {
        rear = NULL;
    }
    free(temp);
    count--;
    printf("\nConsumed: %d", value);
}

int main() {
    int choice, value;
    while (1) {
        printf("\n1. Produce \t 2. Consume \t 3. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("\nEnter the value: ");
                scanf("%d", &value);
                produce(value);
                break;
            case 2:
                consume();
                break;
            case 3:
                printf("\nExiting the program\n");
                return 0;
            default:
                printf("\nInvalid choice, please try again.");
        }
    }
}
```



C:\Users\marya\Downloads\L

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

Enter the value: 10

Produced: 10
1. Produce       2. Consume      3. Exit
Enter your choice: 2

Consumed: 10
1. Produce       2. Consume      3. Exit
Enter your choice: 3

Exiting the program

--------------------------------
Process exited after 11.77 seconds with return value 0
Press any key to continue . . .
```

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

ANSWER: Using a stack instead of an array (queue) in the Producer-Consumer problem changes the order of consumption:

● Queue (FIFO): The oldest item is consumed first.

● Stack (LIFO): The newest item is consumed first.

This can lead to older items being left in the buffer, which may not be ideal for real-time processing. However, it can be useful in scenarios where recent data is prioritized