NAME : TASKEEN SADIQ **ROLL NO : DT-22004**

OS LAB 04

QUESTION :01

```
#include <stdio.h>

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

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

        switch(choice) {
            case 1:
                // Check if the buffer is full
                if((in + 1) % bufsize == out) {
                    printf("\nBuffer is Full\n");
                } else {
                    // Produce: add an item to the buffer
                    printf("\nEnter the value to produce: ");
                    scanf("%d", &produce);
                    buffer[in] = produce;
                    in = (in + 1) % bufsize;  // Circular increment
                }
                break;

            case 2:
                // Check if the buffer is empty
                if(in == out) {
                    printf("\nBuffer is Empty\n");
                } else {
                    // Consume: remove an item from the buffer
                    consume = buffer[out];
                    printf("\nThe consumed value is %d\n", consume);
                    out = (out + 1) % bufsize;  // Circular increment
                }
                break;
```

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

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

    return 0;
}
```

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

Enter the value to produce: 4

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

The consumed value is 4

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

Exiting the program.

--------------------------------
Process exited after 19.22 seconds with return value 0
Press any key to continue . . .
```

NAME : TASKEEN SADIQ **ROLL NO : DT-22004**

QUESTION : 02

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

// Structure for Linked List Node
struct Node {
    int data;
    struct Node* next;
};

// Structure for the Queue (Circular Buffer)
struct Queue {
    struct Node* front;
    struct Node* rear;
    int size;
    int max_size;
};

// Function to create a new node
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}

// Function to initialize the queue
void initQueue(struct Queue* q, int max_size) {
    q->front = NULL;
    q->rear = NULL;
    q->size = 0;
    q->max_size = max_size;
}

// Function to check if the queue is full
int isFull(struct Queue* q) {
    return q->size == q->max_size;
}

// Function to check if the queue is empty
int isEmpty(struct Queue* q) {
    return q->size == 0;
```

```
}

// Function to add an item to the queue (Producer)
void produce(struct Queue* q, int value) {
    if (isFull(q)) {
        printf("\nBuffer is Full! Cannot produce.\n");
    } else {
        struct Node* newNode = createNode(value);
        if (isEmpty(q)) {
            q->front = q->rear = newNode;
        } else {
            q->rear->next = newNode;
            q->rear = newNode;
        }
        q->size++;
        printf("\nProduced: %d\n", value);
    }
}

// Function to remove an item from the queue (Consumer)
void consume(struct Queue* q) {
    if (isEmpty(q)) {
        printf("\nBuffer is Empty! Cannot consume.\n");
    } else {
        struct Node* temp = q->front;
        int consumedValue = temp->data;
        q->front = q->front->next;
        free(temp);
        q->size--;
        printf("\nConsumed: %d\n", consumedValue);
    }
}

// Function to display the current items in the queue
void display(struct Queue* q) {
    if (isEmpty(q)) {
        printf("\nBuffer is Empty\n");
    } else {
        struct Node* temp = q->front;
        printf("\nBuffer contents: ");
        while (temp != NULL) {
            printf("%d ", temp->data);
            temp = temp->next;
        }
```

```
        printf("\n");
    }
}

// Main function
int main() {
    struct Queue q;
    initQueue(&q, 10);  // Set buffer size to 10

    int choice, value;

    while (1) {
        printf("\n1. Produce \t 2. Consume \t 3. Display \t 4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                printf("Enter the value to produce: ");
                scanf("%d", &value);
                produce(&q, value);
                break;

            case 2:
                consume(&q);
                break;

            case 3:
                display(&q);
                break;

            case 4:
                printf("\nExiting the program.\n");
                exit(0);

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

    return 0;
}
```

NAME : TASKEEN SADIQ **ROLL NO : DT-22004**

```
1. Produce      2. Consume      3. Display      4. Exit
Enter your choice: 1
Enter the value to produce: 5

Produced: 5

1. Produce      2. Consume      3. Display      4. Exit
Enter your choice: 2

Consumed: 5

1. Produce      2. Consume      3. Display      4. Exit
Enter your choice: 3

Buffer is Empty

1. Produce      2. Consume      3. Display      4. Exit
Enter your choice: 4

Exiting the program.

--------------------------------
Process exited after 43.1 seconds with return value 0
Press any key to continue . . .
```

NAME : TASKEEN SADIQ **ROLL NO : DT-22004**

QUESTION : 03

In the **producer-consumer problem**, using a **stack** instead of an **array** changes the way items are consumed:

- **Array (FIFO)**: The first item produced is the first one consumed (First In, First Out). This is typical for producer-consumer problems.

- **Stack (LIFO)**: The last item produced is the first one consumed (Last In, First Out). This means items are consumed in reverse order.

Impact:

- **Array**: Items are processed in the order they were produced.
- **Stack**: The most recent item is consumed first, which may not be suitable if the order matters.

Example Scenario:

Let's say we have a **Producer** producing tasks (items), and a **Consumer** that consumes those tasks.

With an Array (FIFO):

Producer adds tasks in the order `Task 1`, `Task 2`, `Task 3`, and the consumer consumes them in the same order (`Task 1`, `Task 2`, `Task 3`).

- **Queue**: Producer adds to the queue (`Task 1 -> Task 2 -> Task 3`).
- Consumer processes the tasks in the same order (`Task 1 -> Task 2 -> Task 3`).

With a Stack (LIFO):

Producer adds tasks in the order `Task 1`, `Task 2`, `Task 3`, but the consumer consumes them in reverse order (`Task 3`, `Task 2`, `Task 1`).

- **Stack**: Producer pushes onto the stack (`Task 1 -> Task 2 -> Task 3`).
- Consumer pops from the stack and consumes in reverse order (`Task 3 -> Task 2 -> Task 1`).