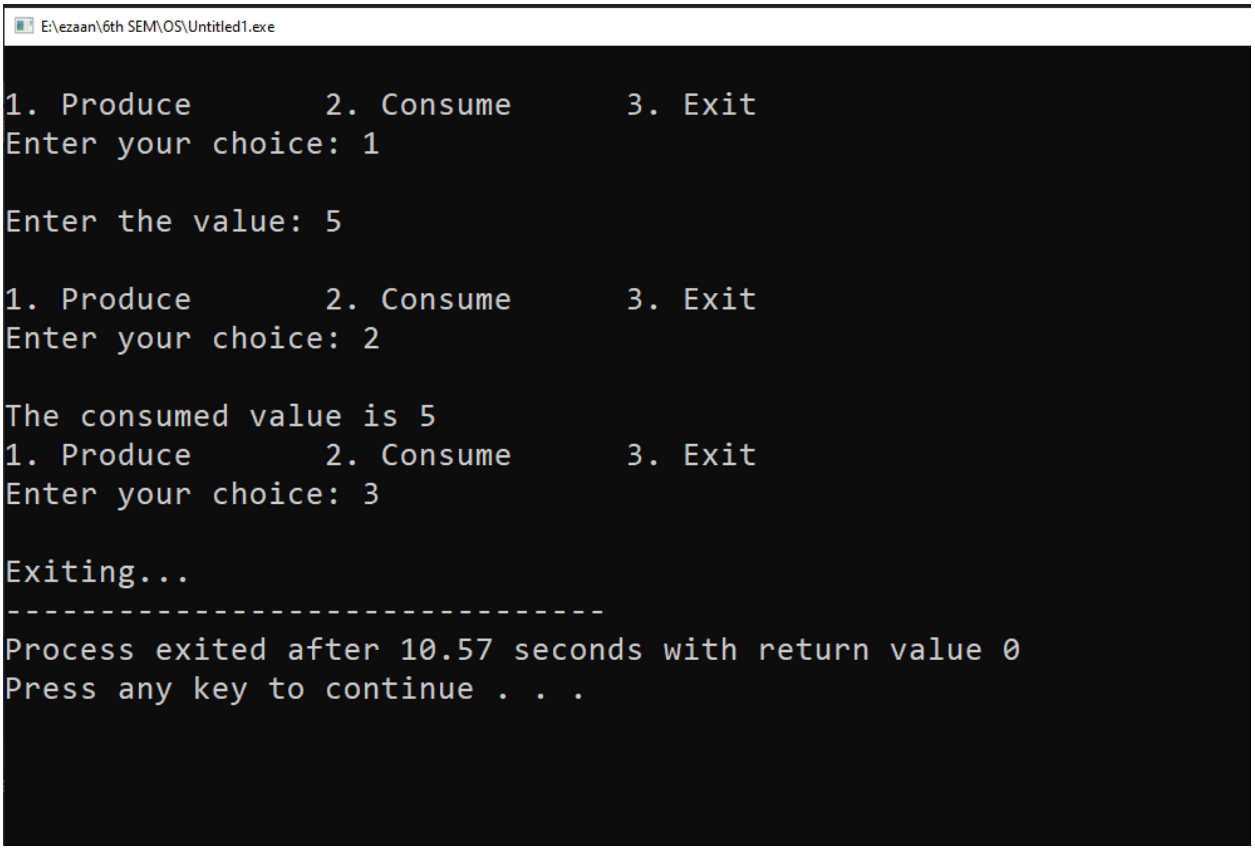
QUESTION # 01:



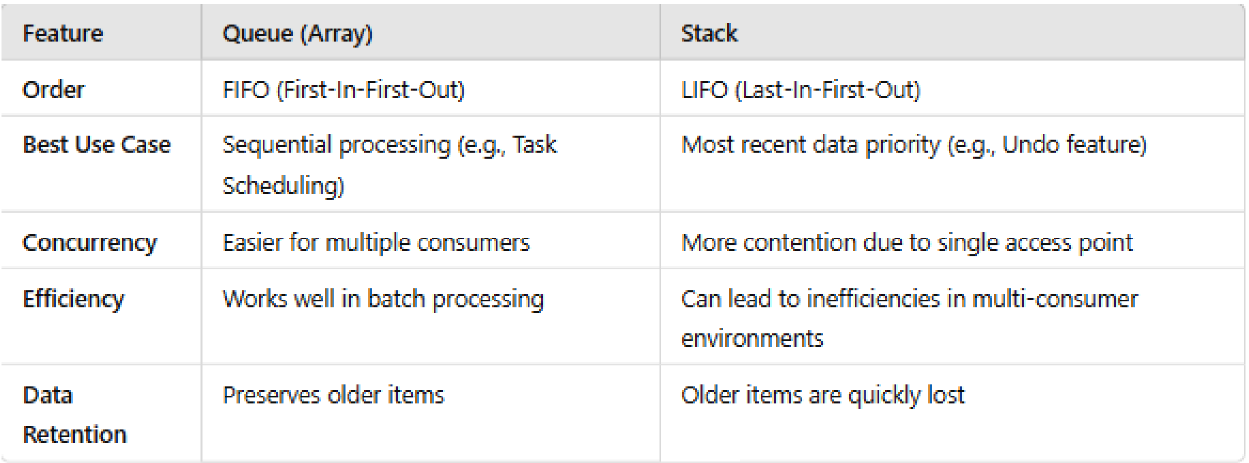
```
1. Produce        2. Consume      3. Exit
Enter your choice: 1

Enter the value: 5

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

The consumed value is 5
1. Produce        2. Consume      3. Exit
Enter your choice: 3

Exiting...
--------------------------------
Process exited after 10.57 seconds with return value 0
Press any key to continue . . .
```

QUESTION # 02:

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

#define BUFFER_SIZE 10

// Node structure for the linked list
typedef struct Node {
    int data;
    struct Node* next;
} Node;

Node* head = NULL; // Head of the linked list
Node* tail = NULL; // Tail of the linked list
int count = 0; // Buffer count
```

```
pthread_mutex_t mutex;
sem_t full, empty;

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

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

int removeItem() {
    if (head == NULL) return -1; // Should never happen

    Node* temp = head;
    int value = temp->data;
    head = head->next;

    if (head == NULL) {
        tail = NULL;
    }

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

void* producer(void* arg) {
    int item;
    while (1) {
        item = rand() % 100; // Produce a random item
        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 = removeItem();
        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(&full, 0, 0);
    sem_init(&empty, 0, BUFFER_SIZE);

    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(&full);
    sem_destroy(&empty);

    return 0;
}
```

QUESTION # 03:



Feature	Queue (Array)	Stack
Order	FIFO (First-In-First-Out)	LIFO (Last-In-First-Out)
Best Use Case	Sequential processing (e.g., Task Scheduling)	Most recent data priority (e.g., Undo feature)
Concurrency	Easier for multiple consumers	More contention due to single access point
Efficiency	Works well in batch processing	Can lead to inefficiencies in multi-consumer environments
Data Retention	Preserves older items	Older items are quickly lost