NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & IT

Specialization in Data Science

CT-353
OPERATING SYSTEMS

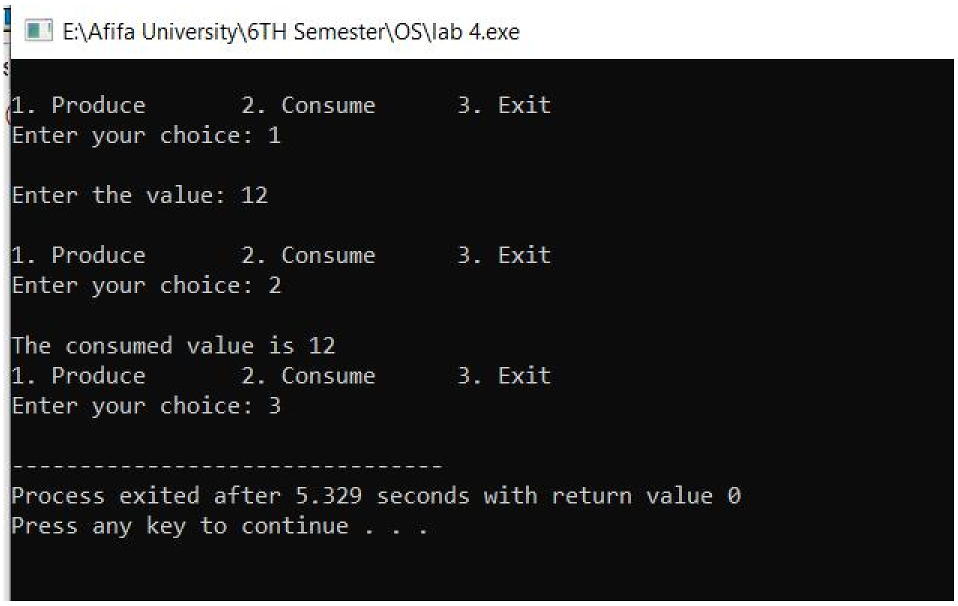
Name : Afifa Siddique
Roll No : DT-22003

Submitted to : Sir Muhammad Abdullah Siddiqui

LAB NO : 04

Q1: Implement the above code and paste the screen shot of the output.

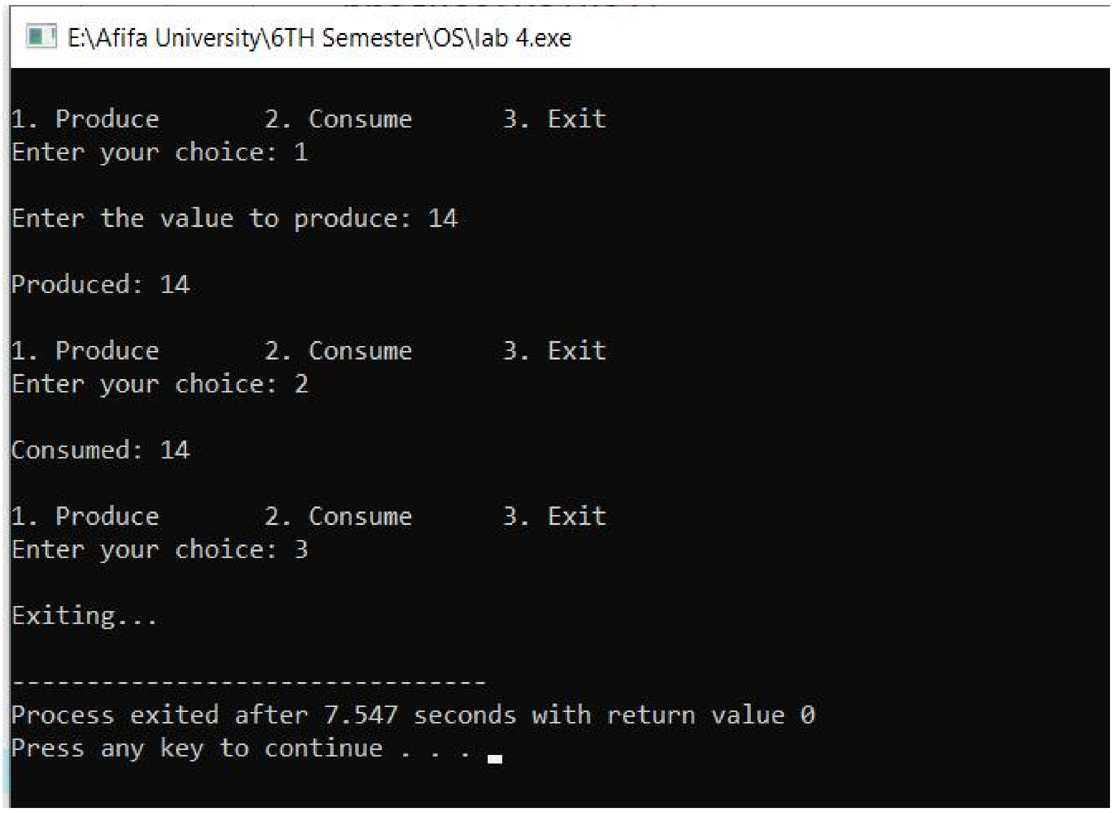
```
#include <stdio.h>

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;
      }
   }
}
```


E:\Afifa University\6TH Semester\OS\lab 4.exe

```
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

--------------------------------
Process exited after 5.329 seconds with return value 0
Press any key to continue . . .
```

Q2: 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.

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

#define MAX_SIZE 10

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

// Create pointers for the front and rear of the queue (buffer)
struct Node* front = NULL;
struct Node* rear = NULL;
int size = 0; // To keep track of the current size of the buffer

// Function to insert an item at the rear (produce)
void produce(int value) {
    if (size == MAX_SIZE) {
        printf("\nBuffer is Full\n");
        return;
    }

    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = value;
    new_node->next = NULL;
    // If the buffer is empty, the new node becomes the front and rear
    if (rear == NULL) {
        front = rear = new_node;
    } else {
        rear->next = new_node;
        rear = new_node;
    }
    size++;
    printf("\nProduced: %d\n", value);
}

// Function to remove an item from the front (consume)
void consume() {
    if (size == 0) {
        printf("\nBuffer is Empty\n");
        return;
    }

    struct Node* temp = front;
    printf("\nConsumed: %d\n", front->data);
    front = front->next;

    // If the buffer becomes empty after the consume operation
    if (front == NULL) {
        rear = NULL;
    }

    free(temp); // Free the consumed node
    size--;
}

// Main function for producer-consumer simulation
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:
            if (size < MAX_SIZE) {
                printf("\nEnter the value to produce: ");
                scanf("%d", &value);
                produce(value);
            } else {
                printf("\nBuffer is Full!\n");
            }
            break;

        case 2:
            consume();
            break;

        case 3:
            printf("\nExiting...\n");
            exit(0);

        default:
            printf("\nInvalid choice! Please enter 1, 2, or 3.\n");
        }
    }

    return 0;
}
```



E:\Afifa University\6TH Semester\OS\lab 4.exe

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

Enter the value to produce: 14

Produced: 14

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

Consumed: 14

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

Exiting...

--------------------------------
Process exited after 7.547 seconds with return value 0
Press any key to continue . . .
```

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

Using a **stack** (LIFO) for the producer-consumer problem means the **consumer** will consume the most recently produced item first, disrupting the order in which items were produced. The producer adds items to the top of the stack, and the consumer removes them from the top.

In contrast, using an **array/queue** (FIFO) ensures the consumer consumes items in the same order they were produced, which is typically desired in most producer-consumer scenarios.

Key Differences:

- **Stack (LIFO):** The consumer gets the most recent item first, useful for backtracking or undo operations.
- **Array/Queue (FIFO):** The consumer consumes items in the same order they were produced, ideal for task scheduling or resource allocation.

Using a stack can cause issues if the order of consumption matters.
