SVKM'S NMIMS

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT& ENGINEERING

(Campus Name)

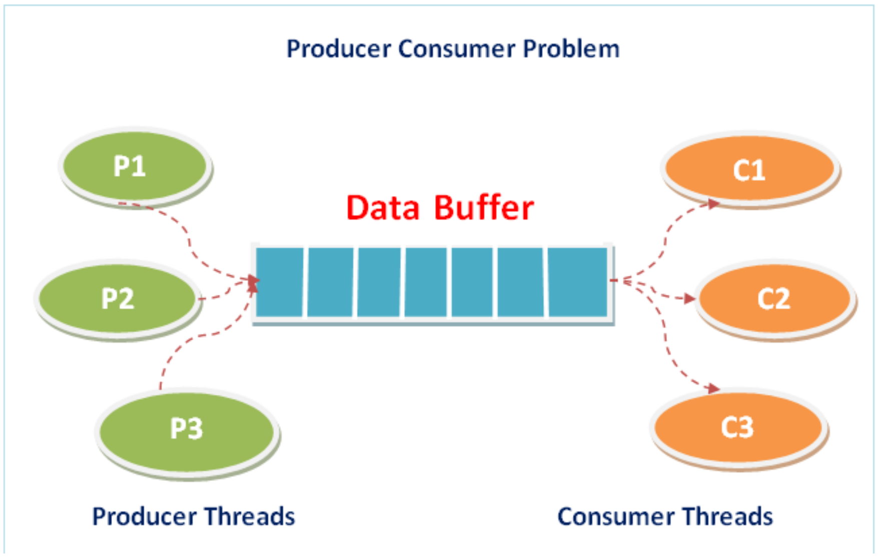
Academic Year: 2022-2023

Practical 5 –Program to demonstrate synchronization through Producer/Consumer problem.

Name:Arjun Mehta	Roll_No:K036	SAP-ID:70102300018

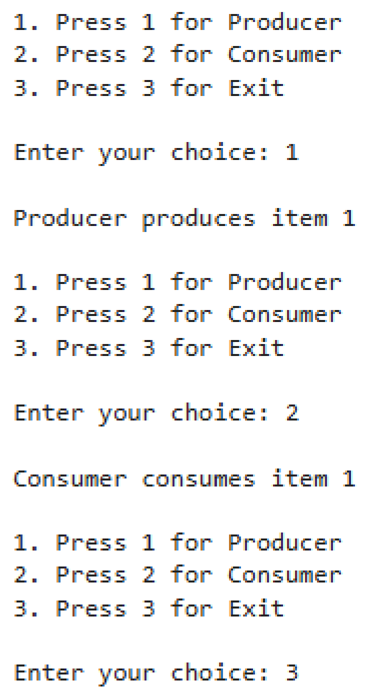
Dear all,

Kindly complete the following task with your name in output file also attach the C/Java program with the file.



Code:

```
#!/usr/bin/env python

# coding: utf-8
# In[7]:


import threading


# In[8]:


# Initialize a mutex to 1
mutex = threading.Lock()

# Number of full slots as 0
full = 0

# Number of empty slots as size of buffer
empty = 10
x = 0


# In[9]:


# Function to produce an item and add it to the buffer
def producer():
    global full, empty, x
    with mutex:
        # Increase the number of full slots by 1
        full += 1
        # Decrease the number of empty slots by 1
        empty -= 1

        # Item produced
        x += 1
        print(f"\nProducer produces item {x}")



# In[10]:



# Function to consume an item and remove it from buffer
def consumer():
    global full, empty, x
    with mutex:
        # Decrease the number of full slots by 1
        full -= 1

        # Increase the number of empty slots by 1
        empty += 1
        print(f"\nConsumer consumes item {x}")
        x -= 1



# In[11]:



# Driver Code
def main():
    while True:
        print("\n1. Press 1 for Producer"
          "\n2. Press 2 for Consumer"
          "\n3. Press 3 for Exit")

        n = int(input("\nEnter your choice: "))

        # Switch Cases
        if n == 1:
            # If mutex is available and empty is non-zero, then it is possible to produce
            if mutex.locked() == False and empty != 0:
                producer()
            else:
                print("Buffer is full!")
        elif n == 2:
            # If mutex is available and full is non-zero, then it is possible to consume
            if mutex.locked() == False and full != 0:
                consumer()
            else:
                print("Buffer is empty!")
        elif n == 3:
            break


# In[12]:


if __name__ == "__main__":
    main()
```



```
1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit

Enter your choice: 1

Producer produces item 1

1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit

Enter your choice: 2

Consumer consumes item 1

1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit

Enter your choice: 3
```

OUTPUT:

Conclusion: -
Write your observation about Producer- consumer problem. How it is more useful in modern operating systems.

References:

studocu.com/row/document/hamdard-university/legal-system/lab-8-producer-consumer-problem/29445188

https://www.geeksforgeeks.org/producer-consumer-problem-in-c/?ref=rp