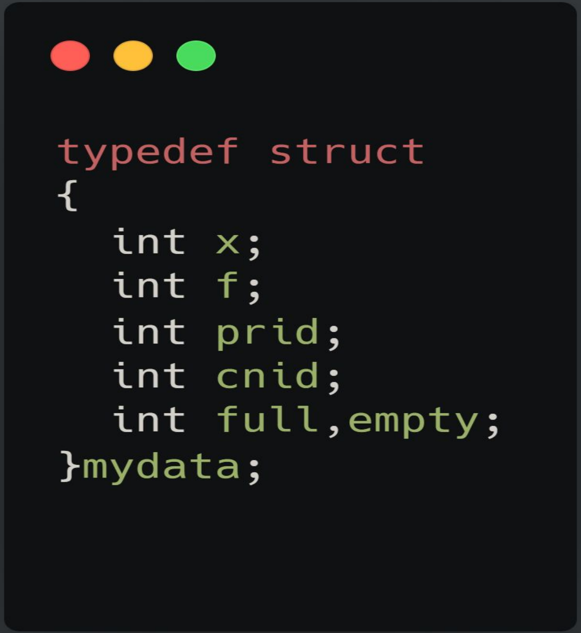
OS Lab Assignment - 3

Question 6

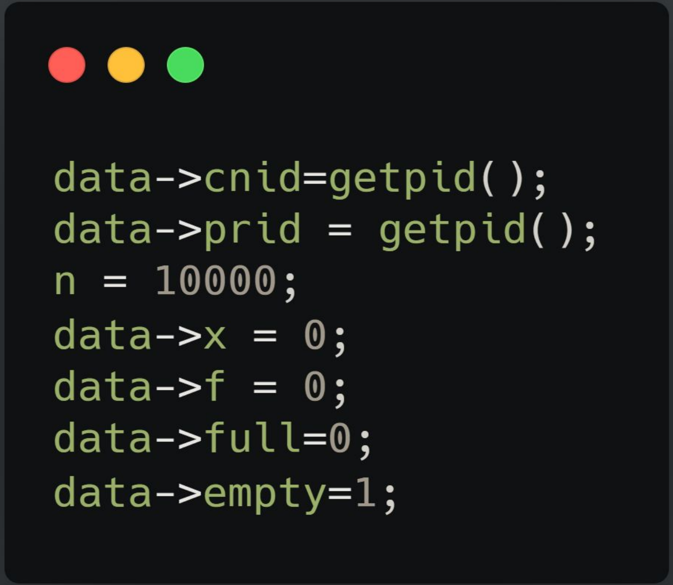
Submitted by:

IIT2019204, IIT2019205, IIT2019208, IIT2019219, IIT2019234

INITIAL DECLARATIONS



```c
typedef struct
{
   int x;
   int f;
   int prid;
   int cnid;
   int full,empty;
}mydata;
```



```c
data->cnid=getpid();
data->prid = getpid();
n = 10000;
data->x = 0;
data->f = 0;
data->full=0;
data->empty=1;
```

USED VARIABLES EXPLANATION

Full - is used as a shared variable between consumer and producer processes which is initialised to 0 in producer process.i.e., **data->full=0;**

Empty - is used as a shared variable between consumer and producer processes which is initialised to 1 in producer process.i.e., **data->empty=1;**

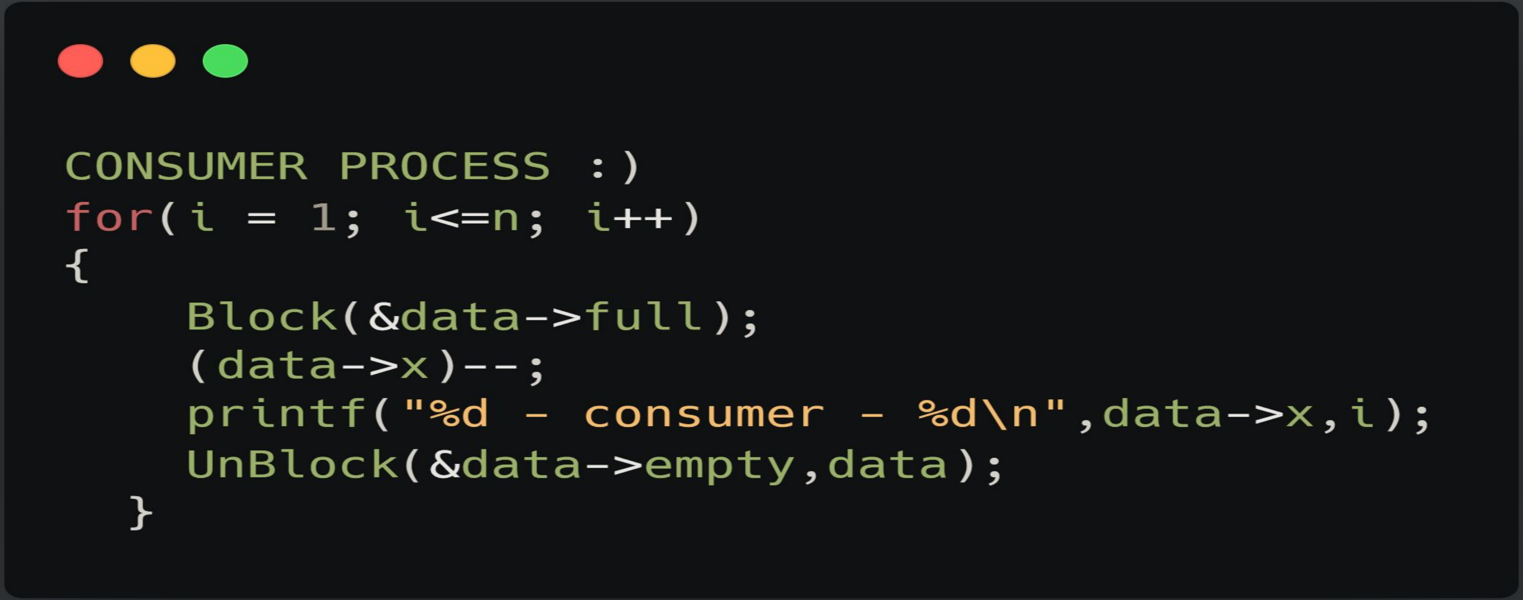
x - a shared variable between consumer and producer processes which increments or decrements. **data->x=0**

prid-is a variable which is used to store id of the producer.**data->prid=getpid()**

cnid-is a variable which is used to store id of the consumer.**data->cnid=getpid()**

n - is number of times each consumer and producer process be executed.

CONSUMER PROCESS :



```
CONSUMER PROCESS :)
for(i = 1; i<=n; i++)
{
    Block(&data->full);
    (data->x)--;
    printf("%d - consumer - %d\n",data->x,i);
    UnBlock(&data->empty,data);
  }
```

CONSUMER PROCESS

The above consumer process is made in such a way that there is no race condition.

CODE EXPLANATION:

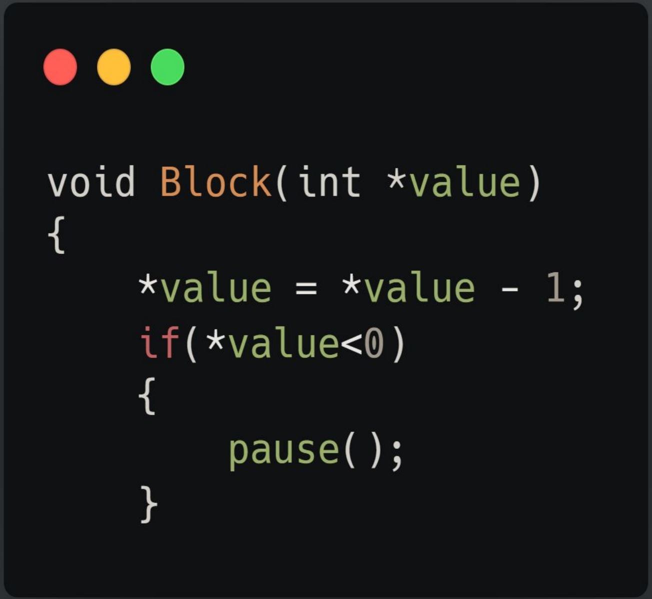
Whenever the process enters the for loop, it calls the function **Block()** which takes address of **data->full** as argument .

If **Block()** function is successfully run , then the process enters critical section i.e., **data->x --**

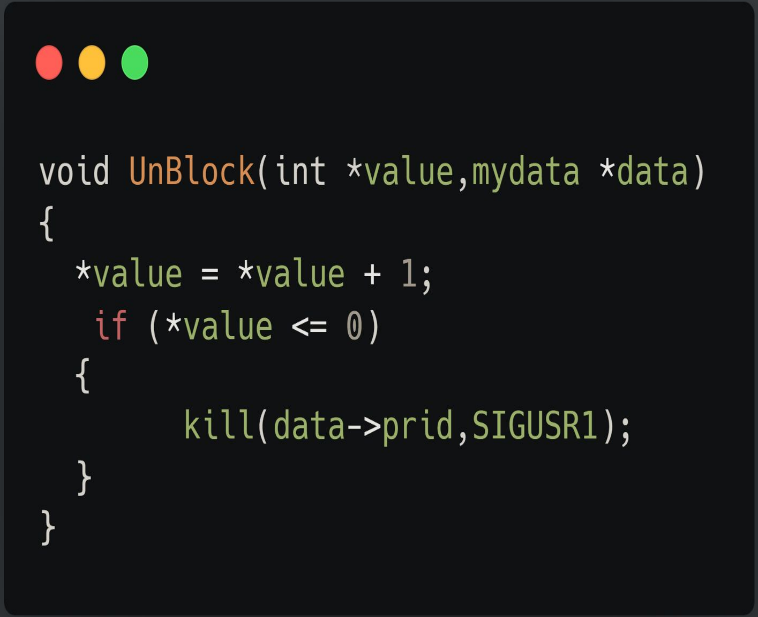
Then the **UnBlock()** is called which takes the address of **data->empty** and the and address of the struct variable as arguments.

BLOCK FUNCTION

UNBLOCK FUNCTION



```c
void Block(int *value)
{
     *value = *value - 1;
     if(*value<0)
     {
          pause();
     }
```



```c
void UnBlock(int *value,mydata *data)
{
  *value = *value + 1;
   if (*value <= 0)
  {
        kill(data->prid,SIGUSR1);
  }
}
```

Explanation for block function in consumer process

=> the image on the left shows the block function and right image shows the unblock function of consumer processes.

=>whenever the the function **Block()** is being called with parameter as address of **data->full(in function it is now val)**, then the value that is stored in address of **data->full** is being decreased by 1, and

=>if the if condition, which checks if value of data->full is less than 0, is true then that particular process pauses, for this **pause()** system call is being used .

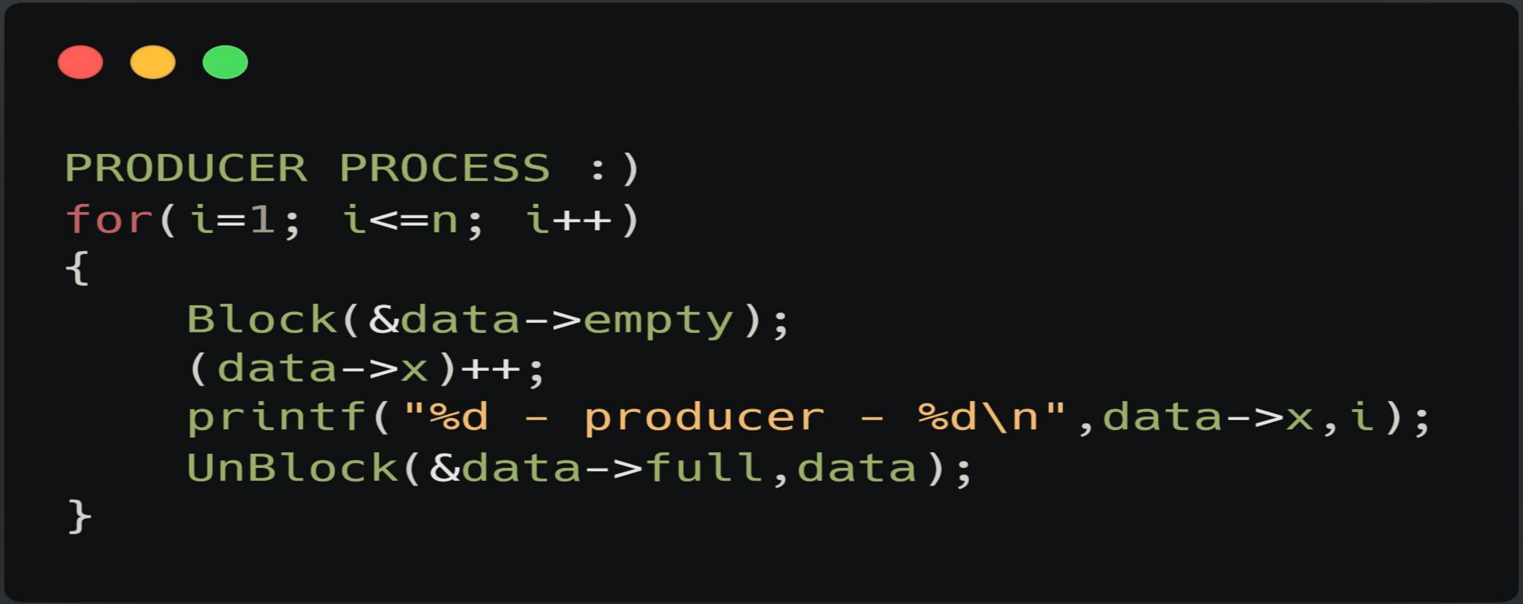
Explanation for unblock function in consumer process

=>**int *value ,mydata *data** - these are the parameters of UnBlock function in both processes and this function return type is void.

=>whenever the the function **UnBlock()** is being called with parameter as address of **data->empty(in function it is val)**,and address of mydata object **data** then the value that is stored in address of **data->empty** is being increased by 1.

=>if the if condition, which checks if value of **data->empty** is less than or equal to 0, is true then that process,then it calls the system call **kill(data->prid,SIGUSR1)** which resumes the paused producer process

PRODUCER PROCESS :



```
PRODUCER PROCESS :)
for(i=1; i<=n; i++)
{
    Block(&data->empty);
    (data->x)++;
    printf("%d - producer - %d\n",data->x,i);
    UnBlock(&data->full,data);
}
```

PRODUCER PROCESS

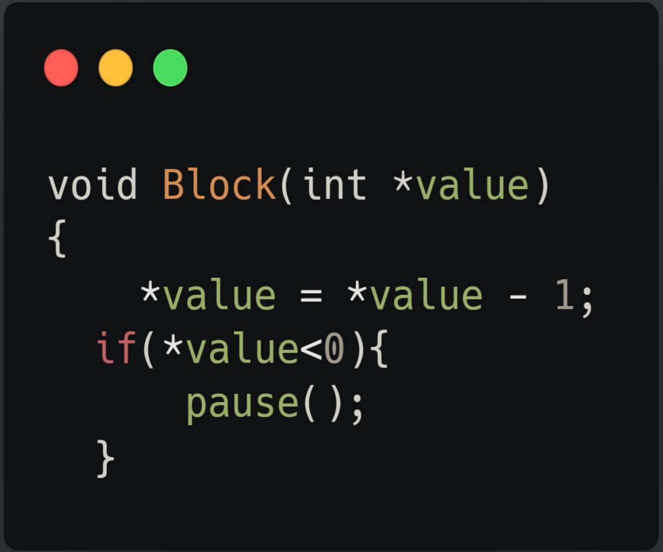
The above producer process is made in such a way that there is no race condition.

CODE EXPLANATION:

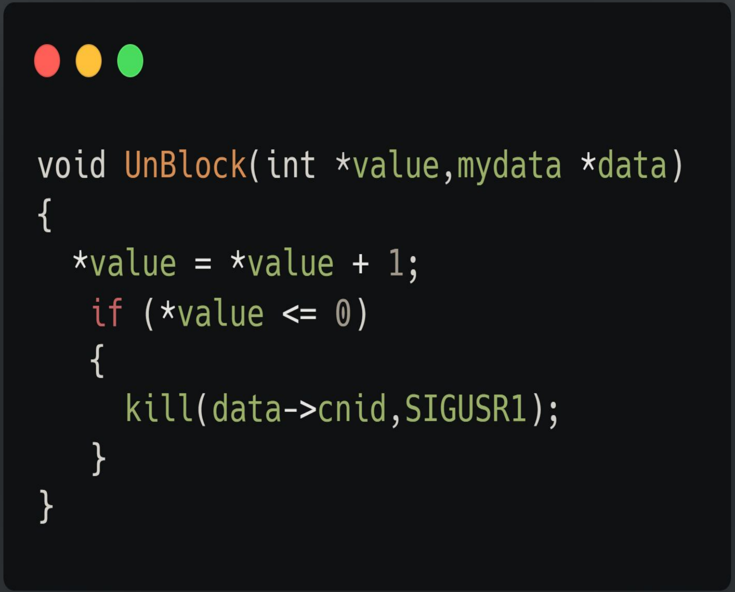
Whenever the process enters the for loop, it calls the function **Block()** which takes address of **data->empty** as argument .

If **Block()** function is successfully run , then the process enters critical section i.e., **data->x ++**

Then the **UnBlock()** is called which takes the address of **data->full** and the address of the struct variable as arguments.



```c
void Block(int *value)
{
    *value = *value - 1;
  if(*value<0){
     pause();
  }
```



```c
void UnBlock(int *value,mydata *data)
{
  *value = *value + 1;
   if (*value <= 0)
   {
     kill(data->cnid,SIGUSR1);
   }
}
```

Explanation of block function in producer process

=> the left image shows the block function and right image shows the unblock function of producer process.

=>whenever the the function **Block()** is being called with parameter as address of **data->full(in function it is now val)**, then the value that is stored in address of **data->full** is being decreased by 1.

=>if the if condition, which checks if value of data->full is less than 0, is true then that particular process pauses where **pause()** system call is being used .

Explanation for unblock function in producer process

=>int *value ,mydata *data-these are the parameters of unblock function in producer processes and this function return type is void.

=>whenever the function **UnBlock()** is being called with parameter as address of **data->empty(in function it is now val)**,and address of mydata object **data** then the value that is stored in address of **data->empty** is being increased by 1.

=>if the if condition, which checks if value of **data->empty** is less than or equal to 0, is true, then it calls the system call **kill(data->cnid,SIGUSR1)** which resumes the paused consumer process

A BRIEF PROCEDURE OF SYNCHRONISATION

=> for the first time when consumer process tries to enter critical section of consumer process, it firstly executes **Block()** function call with **&data->full** as parameter , since, **data->full=0** , the **Block()** function makes ***value=-1(*value<0)** , therefore the process consumer pauses,because of pause() system call.

=>Since, producer process is under execution of its own critical section**(ALREADY THE CONSUMER PROCESS IS BEING PAUSED AS IT CANNOT ENTER CRITICAL SECTION , HERE IT SHOWS MUTUAL EXCLUSION)** , then it executes **Block()** function call with **&data->empty** as argument , since **data->empty=1** , the **Block()** function makes ***value=0(*value>0),** so it comes out of **Block()** function , as usual makes changes in the critical section , then producer process undergoes **UnBlock()** system call , since **data->full=-1** , the function makes **data->full=0(*value++)** and *value <=0.

A BRIEF PROCEDURE OF SYNCHRONISATION

=>so the **kill ()** system call is being called and consumer process is resumed

=>now, both the consumer and producer processes are running .

=>when the consumer process is in critical section then producer process is sleeping (**while(data->f==0)**).**(HERE WE CAN SHOW THAT IF ONE PROCESS IS IN CRITICAL SECTION OTHER PROCESS SHOULD NOT BE IN CRITICAL SECTION i.e., MUTUAL EXCLUSION)**

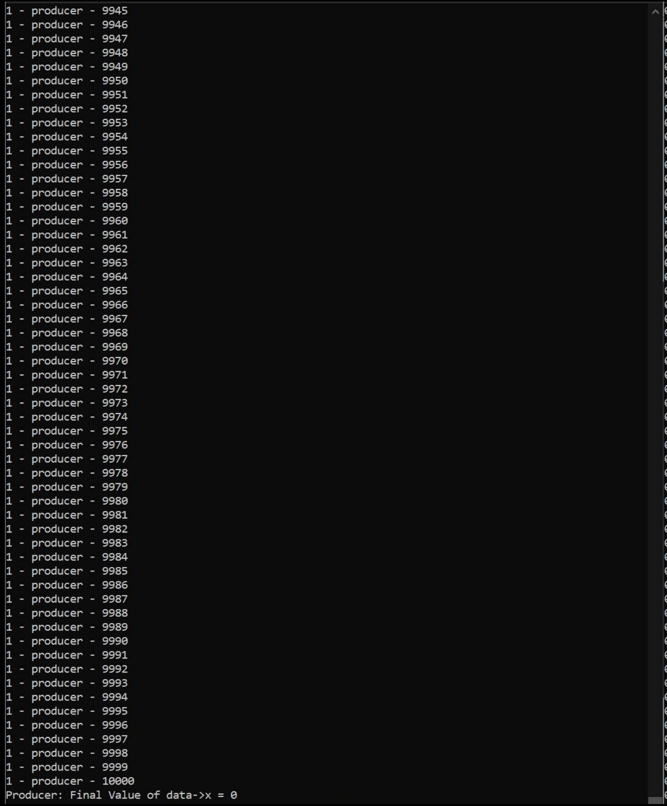
=>when consumer process completes it's critical section and comes out of for loop then **(data->f=1)** is done and producer process comes out of sleep and executes printf statement .

=> Since, the incrementation and decrementation are happening one after the other **data->x** remains 0.(**WHICH SHOWS NO RACE CONDITION)**

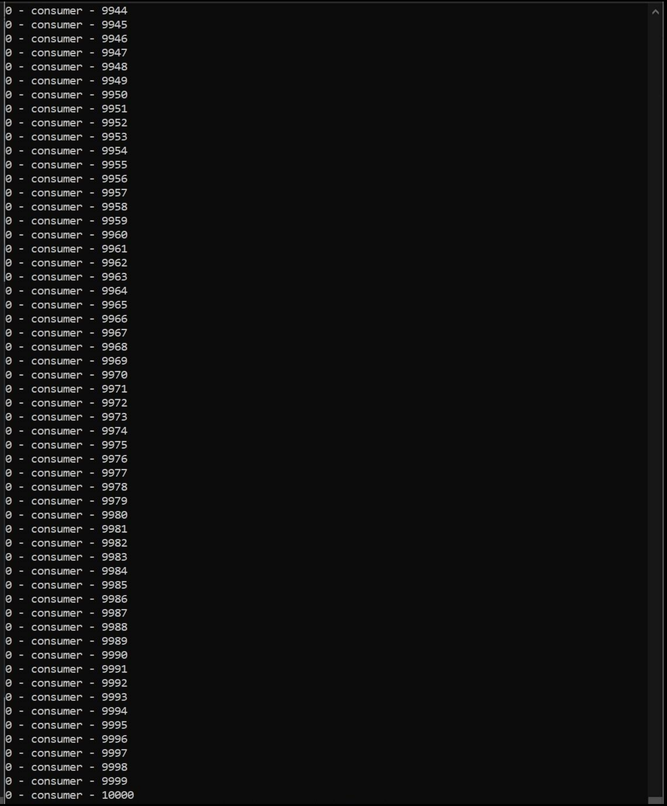
The below 3 example outputs show the the process synchronisation between consumer process and producer process .

For n=100,n=1000,n=10000.

Example 1 - n = 10000

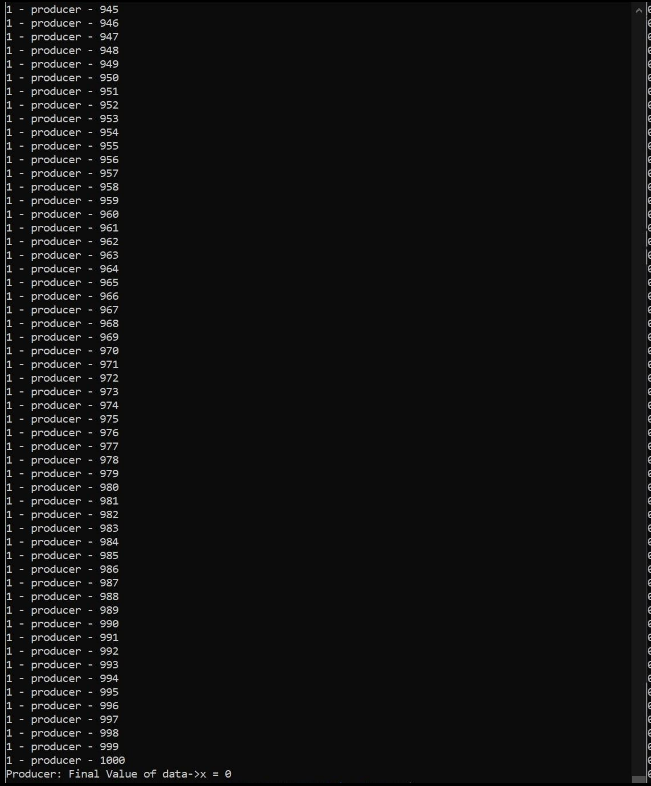


```
1 - producer - 9945
1 - producer - 9946
1 - producer - 9947
1 - producer - 9948
1 - producer - 9949
1 - producer - 9950
1 - producer - 9951
1 - producer - 9952
1 - producer - 9953
1 - producer - 9954
1 - producer - 9955
1 - producer - 9956
1 - producer - 9957
1 - producer - 9958
1 - producer - 9959
1 - producer - 9960
1 - producer - 9961
1 - producer - 9962
1 - producer - 9963
1 - producer - 9964
1 - producer - 9965
1 - producer - 9966
1 - producer - 9967
1 - producer - 9968
1 - producer - 9969
1 - producer - 9970
1 - producer - 9971
1 - producer - 9972
1 - producer - 9973
1 - producer - 9974
1 - producer - 9975
1 - producer - 9976
1 - producer - 9977
1 - producer - 9978
1 - producer - 9979
1 - producer - 9980
1 - producer - 9981
1 - producer - 9982
1 - producer - 9983
1 - producer - 9984
1 - producer - 9985
1 - producer - 9986
1 - producer - 9987
1 - producer - 9988
1 - producer - 9989
1 - producer - 9990
1 - producer - 9991
1 - producer - 9992
1 - producer - 9993
1 - producer - 9994
1 - producer - 9995
1 - producer - 9996
1 - producer - 9997
1 - producer - 9998
1 - producer - 9999
1 - producer - 10000
Producer: Final Value of data->x = 0
```

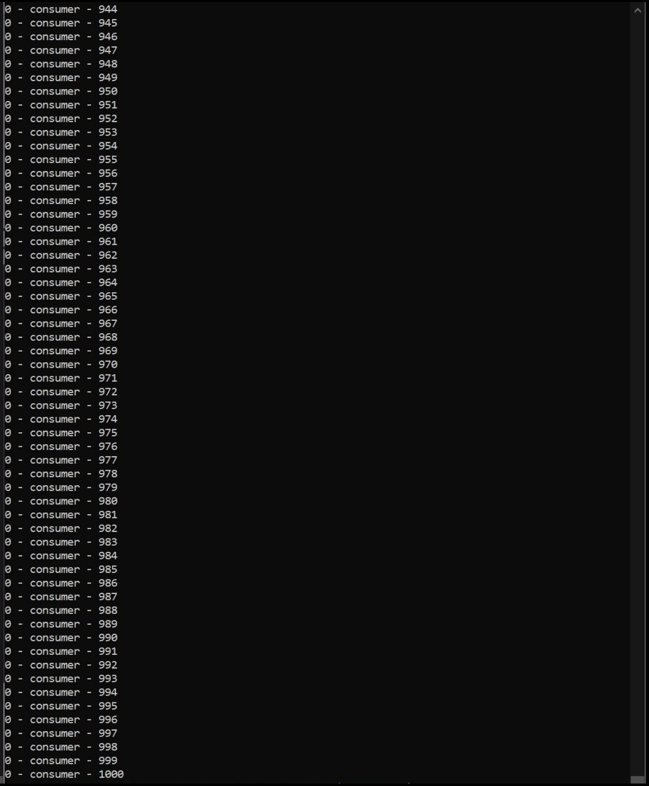


```
0 - consumer - 9944
0 - consumer - 9945
0 - consumer - 9946
0 - consumer - 9947
0 - consumer - 9948
0 - consumer - 9949
0 - consumer - 9950
0 - consumer - 9951
0 - consumer - 9952
0 - consumer - 9953
0 - consumer - 9954
0 - consumer - 9955
0 - consumer - 9956
0 - consumer - 9957
0 - consumer - 9958
0 - consumer - 9959
0 - consumer - 9960
0 - consumer - 9961
0 - consumer - 9962
0 - consumer - 9963
0 - consumer - 9964
0 - consumer - 9965
0 - consumer - 9966
0 - consumer - 9967
0 - consumer - 9968
0 - consumer - 9969
0 - consumer - 9970
0 - consumer - 9971
0 - consumer - 9972
0 - consumer - 9973
0 - consumer - 9974
0 - consumer - 9975
0 - consumer - 9976
0 - consumer - 9977
0 - consumer - 9978
0 - consumer - 9979
0 - consumer - 9980
0 - consumer - 9981
0 - consumer - 9982
0 - consumer - 9983
0 - consumer - 9984
0 - consumer - 9985
0 - consumer - 9986
0 - consumer - 9987
0 - consumer - 9988
0 - consumer - 9989
0 - consumer - 9990
0 - consumer - 9991
0 - consumer - 9992
0 - consumer - 9993
0 - consumer - 9994
0 - consumer - 9995
0 - consumer - 9996
0 - consumer - 9997
0 - consumer - 9998
0 - consumer - 9999
0 - consumer - 10000
```

Example 2 - n = 1000

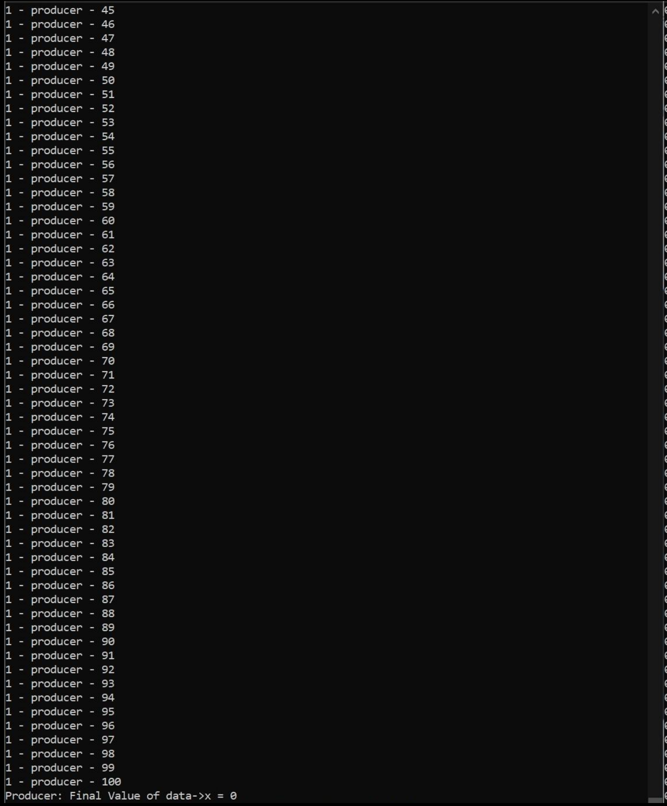


```
1 - producer - 945
1 - producer - 946
1 - producer - 947
1 - producer - 948
1 - producer - 949
1 - producer - 950
1 - producer - 951
1 - producer - 952
1 - producer - 953
1 - producer - 954
1 - producer - 955
1 - producer - 956
1 - producer - 957
1 - producer - 958
1 - producer - 959
1 - producer - 960
1 - producer - 961
1 - producer - 962
1 - producer - 963
1 - producer - 964
1 - producer - 965
1 - producer - 966
1 - producer - 967
1 - producer - 968
1 - producer - 969
1 - producer - 970
1 - producer - 971
1 - producer - 972
1 - producer - 973
1 - producer - 974
1 - producer - 975
1 - producer - 976
1 - producer - 977
1 - producer - 978
1 - producer - 979
1 - producer - 980
1 - producer - 981
1 - producer - 982
1 - producer - 983
1 - producer - 984
1 - producer - 985
1 - producer - 986
1 - producer - 987
1 - producer - 988
1 - producer - 989
1 - producer - 990
1 - producer - 991
1 - producer - 992
1 - producer - 993
1 - producer - 994
1 - producer - 995
1 - producer - 996
1 - producer - 997
1 - producer - 998
1 - producer - 999
1 - producer - 1000
Producer: Final Value of data->x = 0
```

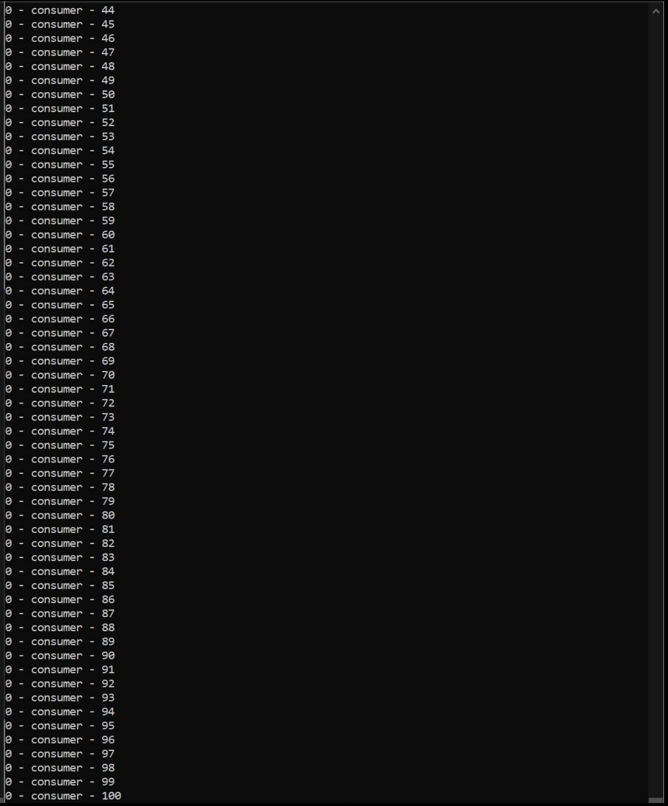


```
0 - consumer - 944
0 - consumer - 945
0 - consumer - 946
0 - consumer - 947
0 - consumer - 948
0 - consumer - 949
0 - consumer - 950
0 - consumer - 951
0 - consumer - 952
0 - consumer - 953
0 - consumer - 954
0 - consumer - 955
0 - consumer - 956
0 - consumer - 957
0 - consumer - 958
0 - consumer - 959
0 - consumer - 960
0 - consumer - 961
0 - consumer - 962
0 - consumer - 963
0 - consumer - 964
0 - consumer - 965
0 - consumer - 966
0 - consumer - 967
0 - consumer - 968
0 - consumer - 969
0 - consumer - 970
0 - consumer - 971
0 - consumer - 972
0 - consumer - 973
0 - consumer - 974
0 - consumer - 975
0 - consumer - 976
0 - consumer - 977
0 - consumer - 978
0 - consumer - 979
0 - consumer - 980
0 - consumer - 981
0 - consumer - 982
0 - consumer - 983
0 - consumer - 984
0 - consumer - 985
0 - consumer - 986
0 - consumer - 987
0 - consumer - 988
0 - consumer - 989
0 - consumer - 990
0 - consumer - 991
0 - consumer - 992
0 - consumer - 993
0 - consumer - 994
0 - consumer - 995
0 - consumer - 996
0 - consumer - 997
0 - consumer - 998
0 - consumer - 999
0 - consumer - 1000
```

Example 3 - n = 100

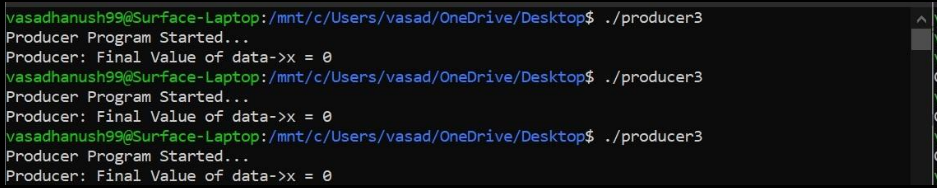


```
1 - producer - 45
1 - producer - 46
1 - producer - 47
1 - producer - 48
1 - producer - 49
1 - producer - 50
1 - producer - 51
1 - producer - 52
1 - producer - 53
1 - producer - 54
1 - producer - 55
1 - producer - 56
1 - producer - 57
1 - producer - 58
1 - producer - 59
1 - producer - 60
1 - producer - 61
1 - producer - 62
1 - producer - 63
1 - producer - 64
1 - producer - 65
1 - producer - 66
1 - producer - 67
1 - producer - 68
1 - producer - 69
1 - producer - 70
1 - producer - 71
1 - producer - 72
1 - producer - 73
1 - producer - 74
1 - producer - 75
1 - producer - 76
1 - producer - 77
1 - producer - 78
1 - producer - 79
1 - producer - 80
1 - producer - 81
1 - producer - 82
1 - producer - 83
1 - producer - 84
1 - producer - 85
1 - producer - 86
1 - producer - 87
1 - producer - 88
1 - producer - 89
1 - producer - 90
1 - producer - 91
1 - producer - 92
1 - producer - 93
1 - producer - 94
1 - producer - 95
1 - producer - 96
1 - producer - 97
1 - producer - 98
1 - producer - 99
1 - producer - 100
Producer: Final Value of data->x = 0
```

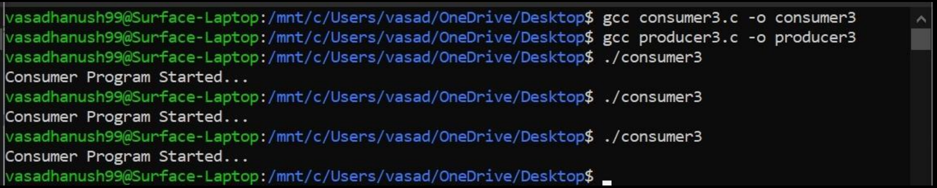


```
0 - consumer - 44
0 - consumer - 45
0 - consumer - 46
0 - consumer - 47
0 - consumer - 48
0 - consumer - 49
0 - consumer - 50
0 - consumer - 51
0 - consumer - 52
0 - consumer - 53
0 - consumer - 54
0 - consumer - 55
0 - consumer - 56
0 - consumer - 57
0 - consumer - 58
0 - consumer - 59
0 - consumer - 60
0 - consumer - 61
0 - consumer - 62
0 - consumer - 63
0 - consumer - 64
0 - consumer - 65
0 - consumer - 66
0 - consumer - 67
0 - consumer - 68
0 - consumer - 69
0 - consumer - 70
0 - consumer - 71
0 - consumer - 72
0 - consumer - 73
0 - consumer - 74
0 - consumer - 75
0 - consumer - 76
0 - consumer - 77
0 - consumer - 78
0 - consumer - 79
0 - consumer - 80
0 - consumer - 81
0 - consumer - 82
0 - consumer - 83
0 - consumer - 84
0 - consumer - 85
0 - consumer - 86
0 - consumer - 87
0 - consumer - 88
0 - consumer - 89
0 - consumer - 90
0 - consumer - 91
0 - consumer - 92
0 - consumer - 93
0 - consumer - 94
0 - consumer - 95
0 - consumer - 96
0 - consumer - 97
0 - consumer - 98
0 - consumer - 99
0 - consumer - 100
```

Example 4 - n = 10000

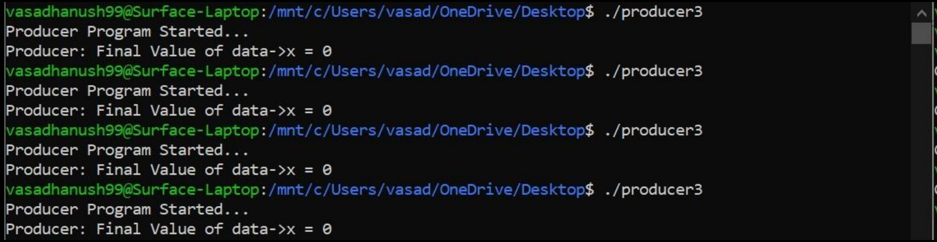


```
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
```

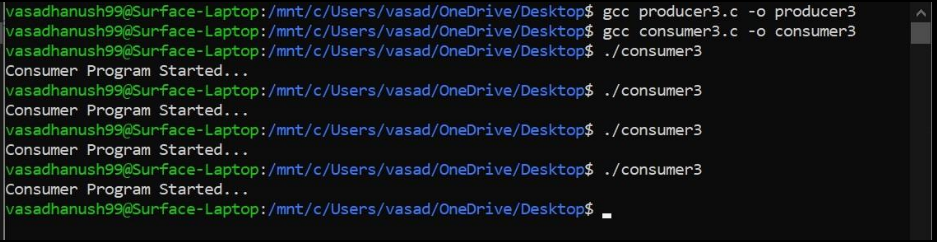


```
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ gcc consumer3.c -o consumer3
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ gcc producer3.c -o producer3
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$
```

Example 5 - n = 1000

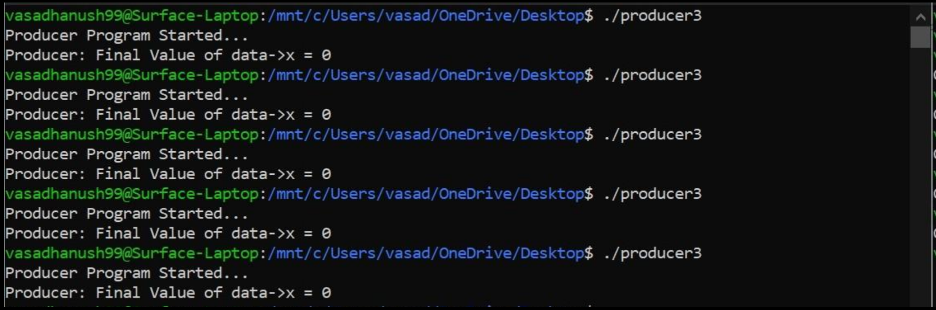


```
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
```

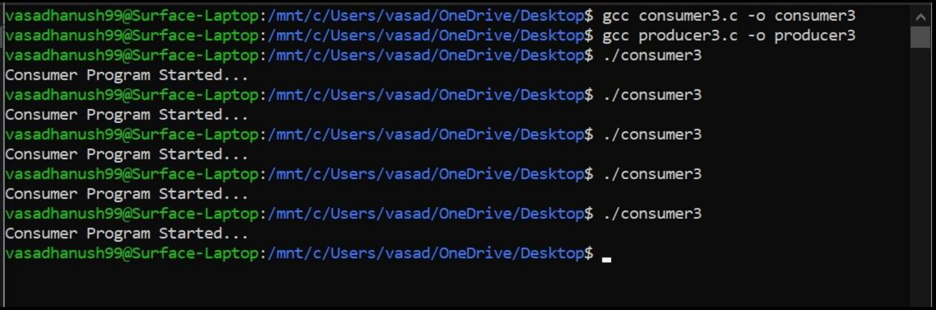


```
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ gcc producer3.c -o producer3
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ gcc consumer3.c -o consumer3
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$
```

Example 6 - n = 100

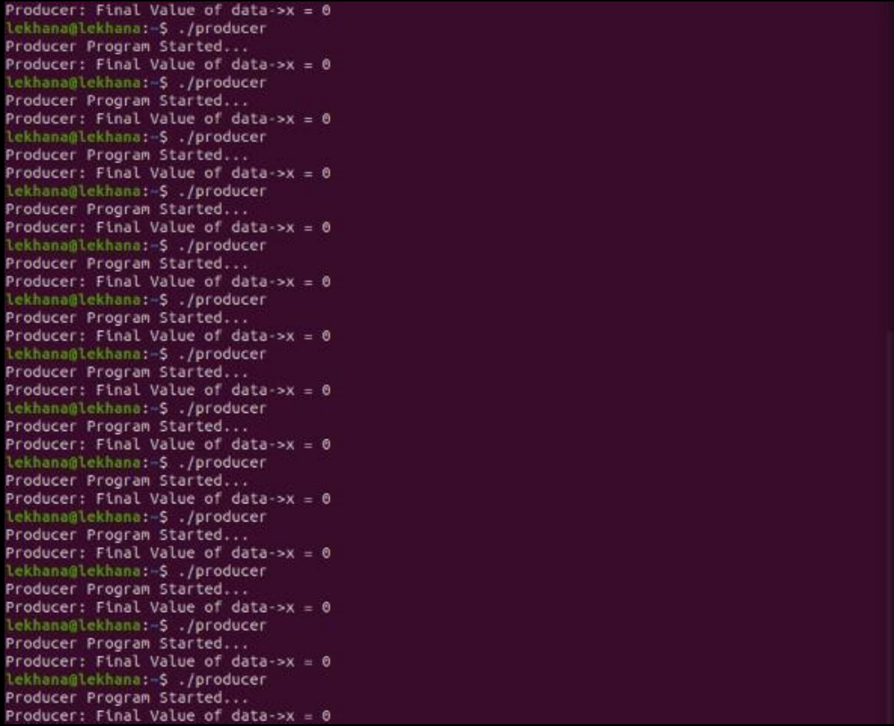


```
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./producer3
Producer Program Started...
Producer: Final Value of data->x = 0
```

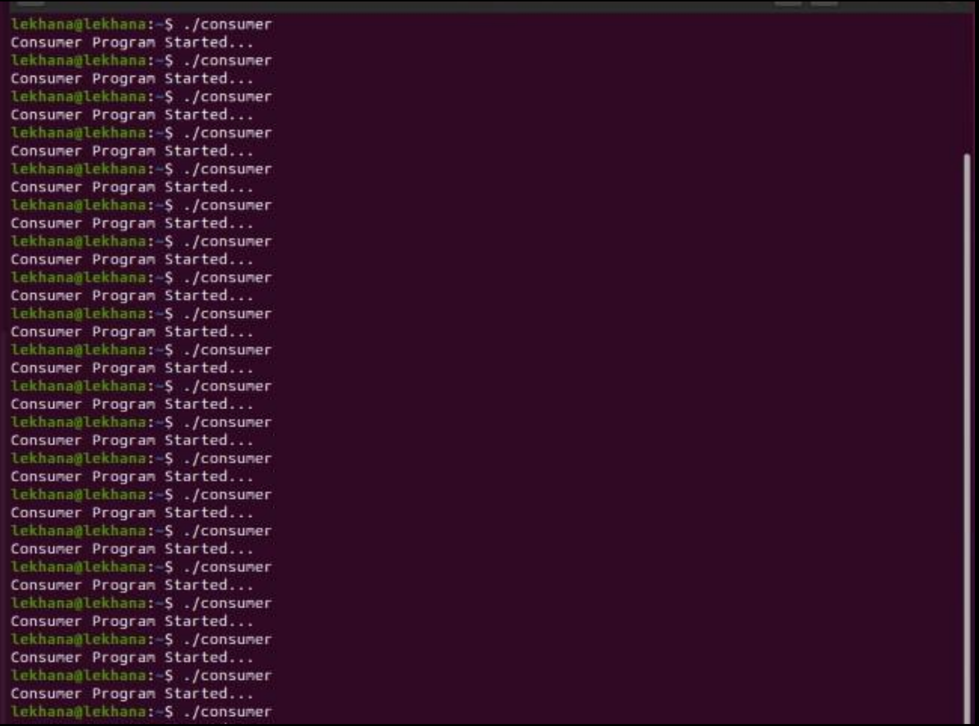


```
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ gcc consumer3.c -o consumer3
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ gcc producer3.c -o producer3
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$ ./consumer3
Consumer Program Started...
vasadhanush99@Surface-Laptop:/mnt/c/Users/vasad/OneDrive/Desktop$
```

Example 7



```
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
lekhana@lekhana:~$ ./producer
Producer Program Started...
Producer: Final Value of data->x = 0
```



```
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
Consumer Program Started...
lekhana@lekhana:~$ ./consumer
```