**Assignment 2**

**Report**

**Q1) Does your program output any garbage? If yes, why?**

**Answer:** No, my program doesn’t output any garbage values. It only displays the value produced and consumed.

**Q 2) Are all the produced values getting consumed? Check your program for a small count like 20.**

**Answer:** No, all the values that are being produced are not always getting consumed by the consumer. I have checked it for smaller value count as well as larger value count. This is because of race condition occurring in the critical section of the process synchronization program. This can be solved by using semaphore in our program so that at a time only one process can make use of shared resource and also semaphore checks for the empty and full state of the buffer.

**Functions:-**

**1 ) Producer function:-**

**In this function, it iterates over the loop for ‘count’ number of times and increments the shared variable’s value by 1 and prints it’s value.**

#include <xinu.h>

#include <prodcons.h>

void producer(int count) {

// Iterates from 0 to count, setting

// the value of the global variable 'n'

// each time.

//print produced value e.g. produced : 8

int i;

for (i = 1 ; i <= count; i++){

n++;

kprintf("produced: %d\n", n);

}

}

**2 ) Consumer Function:-**

**In this function, it iterates over the loop for ‘count’ number of times and reads the shared variable’s value and prints it’s value.**

#include <xinu.h>

#include <prodcons.h>

void consumer(int count) {

// reads the value of the global variable 'n'

// 'count' times.

// print consumed value e.g. consumed : 8

int i;

for (i = 1; i <= count; i++){

kprintf("consumed: %d\n", n);

}

}

**3) Xinu Shell Command (xsh\_prodcons.c):-**

#include <xinu.h>

#include <prodcons.h>

#include <stdio.h>

#include <stddef.h>

#include <stdlib.h>

int n; //Definition for global variable 'n'

/\*Now global variable n will be on Heap so it is accessible all the processes i.e. consume and produce\*/

shellcmd xsh\_prodcons(int nargs, char \*args[])

{

//Argument verifications and validations

int count = 2000; //local varible to hold count

if (nargs == 2){

count = atoi(args[1]);

}

if (count <= 0){

printf("Count value not valid.Please enter positive integer numbers.", count);

return 0;

}

//check args[1] if present assign value to count

//create the process producer and consumer and put them in ready queue.

//Look at the definations of function create and resume in the system folder for reference.

resume( create(producer, 1024, 20, "producer", 1, count));

resume( create(consumer, 1024, 20, "consumer", 1, count));

return (0);

}

**4 ) Header file (prodcons.h):-**

/\*Global variable for producer consumer\*/

extern int n; /\*this is just declaration\*/

/\*function Prototype\*/

void consumer(int count);

void producer(int count);