Software Engineering (CS401)

Lab Assignment 4

**U19CS012**

Q1.) Write a **Program** to create a process that prints **“Hello World”.** Use run in init process to instantiate it and \_pid to print the ids of all create processes.

**Code**

*/\* A "Hello World" Promela model for SPIN. \*/*

proctype Hello()

{

    printf("[Inside] Hello() Process \n");

    printf("Hello() pid : %d \n", \_pid);

}

init

{

    printf("[Inside] init() Process \n");

    printf("init() pid : %d \n", \_pid);

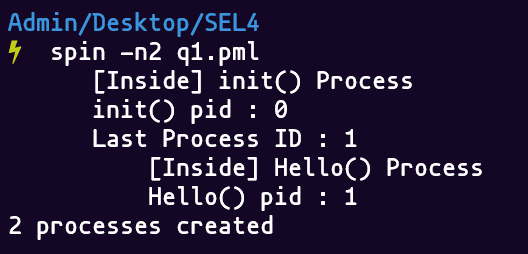
    int lastpid=-1;

    lastpid = run Hello();

    printf("Last Process ID : %d \n", lastpid);

}

**Output**



Q2.) Model **Euclid’s algorithm** for **Greatest Common Divisor**.

**Code**

*/\* Euclid GCD algorithm Implementation. \*/*

proctype gcd(int x;int y)

{

*if*

    :: (y == 0) -> printf("%d\n",x);

    :: (y != 0) -> run gcd(y, x % y)

*fi*

}

init

{

    int number1=12319; *// 12319 = 97\*127*

    int number2=21631; *// 21631 = 97\*223*

    printf("gcd(%d, %d) = ",number1,number2);

    run gcd(number1, number2);

}

Approach 2

*/\* Approach 2 : TC is Higher\*/*

init

{

    int x = 12319;

    int y = 21631;

*do*

    :: x > y -> x = x - y;

    :: y > x -> y = x - y;

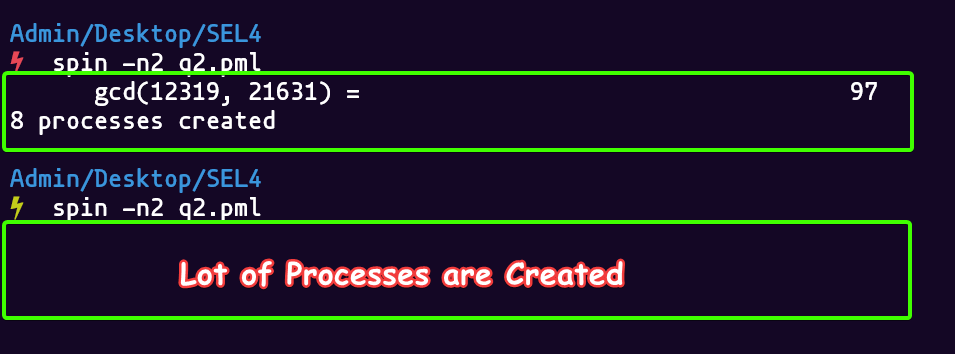
    :: x == y -> *break*;

*od*

    printf("gcd(%d, %d) : %d\n",x,y, x);

}

**Output**



Q3.) Create a process **factorial(n, c)** that recursively computes the factorial of a given non-negative integer “n”.

**Code**

int res = 1;

proctype fac(int n) {

*if*

    :: (n == 0 || n == 1) -> printf("%d\n", res)

    :: (n >=2 ) -> res = res \* n; run fac(n-1)

*fi*

}

init

{

*// Replace with Number whose Factorial Needs to be Found*

    int number = 10;

    printf("%d! = ",number);

    run fac(number);

}

**Output**



Q4.) Create a **Promela** model for Producer-Consumer problem with **buffer size 5**.

* In the **producer-consumer** problem, there is **one Producer** that is producing something and there is **one Consumer** that is consuming the products produced by the Producer.
* The producers and consumers share the same memory buffer that is of fixed-size.
* The **job of the Producer** is to generate the data, put it into the buffer, and **again start generating data**. While the **job of the Consumer** is to consume the data from the buffer.

**Problems that might occur in the Producer-Consumer**

* The producer should produce data only when the buffer is not full. If the buffer is full, then the producer shouldn't be allowed to put any data into the buffer.
* The consumer should consume data only when the buffer is not empty. If the buffer is empty, then the consumer shouldn't be allowed to take any data from the buffer.
* The producer and consumer should not access the buffer at the same time.

**Code**

int SIZE = 5;

int FULL = 0;

int S = 1;

int IN = 0;

int OUT = 0;

byte BUFFER[SIZE];

init {

  printf("Hello");

  BUFFER[0] = ' ';

  BUFFER[1] = ' ';

  BUFFER[2] = ' ';

  BUFFER[3] = ' ';

  BUFFER[4] = ' ';

  run producer();

  run consumer();

  run consumer();

}

proctype consumer() {

*do*

  :: printf("Consumer start\n");

  (FULL > 0) -> FULL = FULL - 1;

  (S == 1) -> S = 0;

  BUFFER[OUT] = ' ';

  OUT = OUT + 1;

  OUT = OUT % SIZE;

  S = 1;

  printf("Buffer: [%c, %c, %c, %c, %c]\n", BUFFER[0], BUFFER[1], BUFFER[2], BUFFER[3], BUFFER[4])

*od*

}

proctype producer() {

*do*

  :: printf("Producer start\n");

  (FULL < SIZE) -> FULL = FULL + 1;

  (S == 1) -> S = 0;

  BUFFER[IN] = '1';

  IN = IN + 1;

  IN = IN % SIZE;

  S = 1;

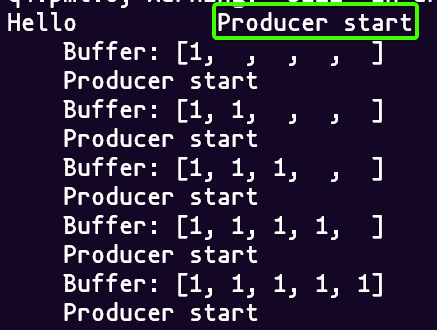
  printf("Buffer: [%c, %c, %c, %c, %c]\n", BUFFER[0], BUFFER[1], BUFFER[2], BUFFER[3], BUFFER[4])

*od*

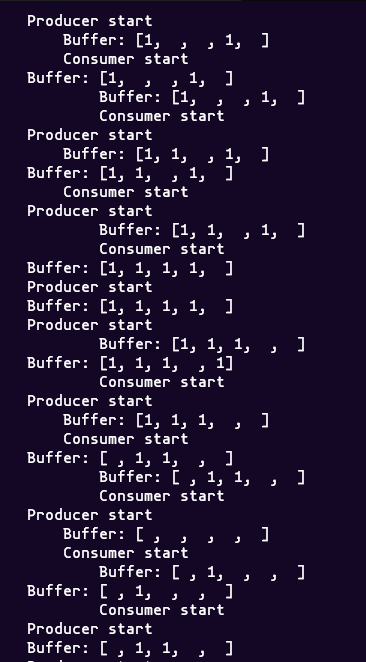
}

**Output**

(Only Producer)



(Both Producer & Consumer)



**SUBMITTED BY**: U19CS012

BHAGYA VINOD RANA