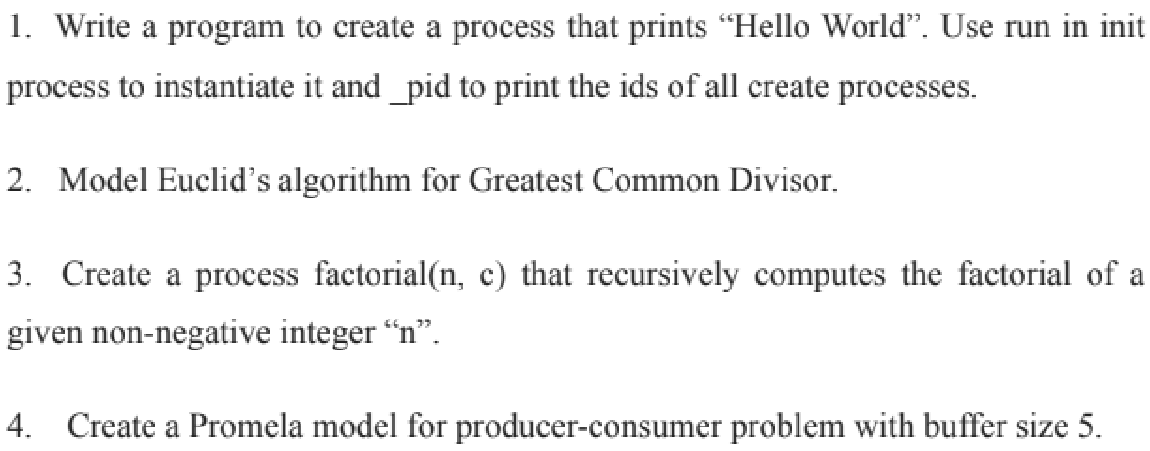
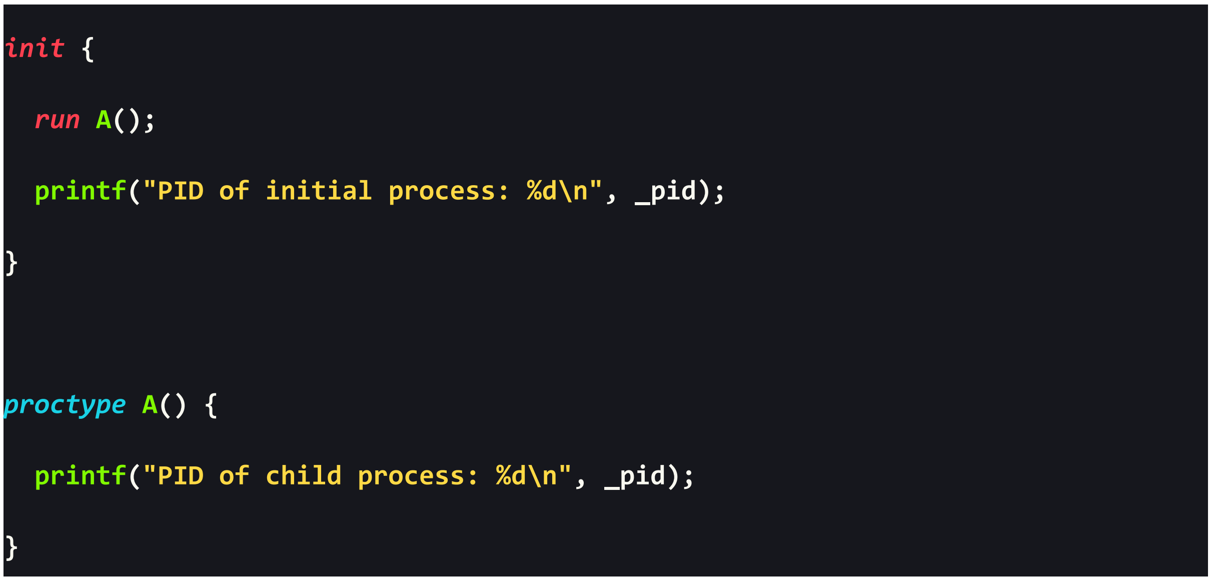
SE LAB 4

SAHIL BONDRE: U18CO021

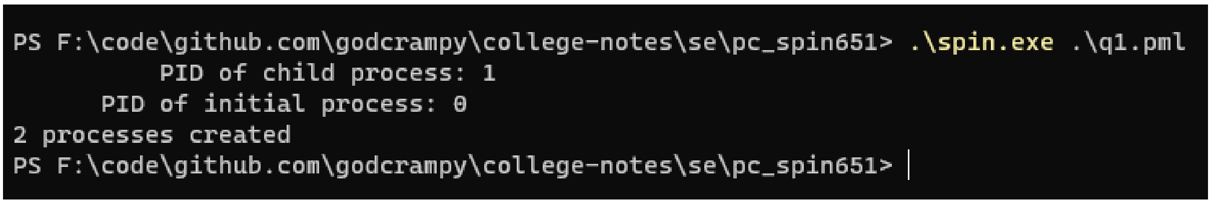


1. Write a program to create a process that prints “Hello World”. Use run in init process to instantiate it and _pid to print the ids of all create processes.
2. Model Euclid’s algorithm for Greatest Common Divisor.
3. Create a process factorial(n, c) that recursively computes the factorial of a given non-negative integer “n”.
4. Create a Promela model for producer-consumer problem with buffer size 5.

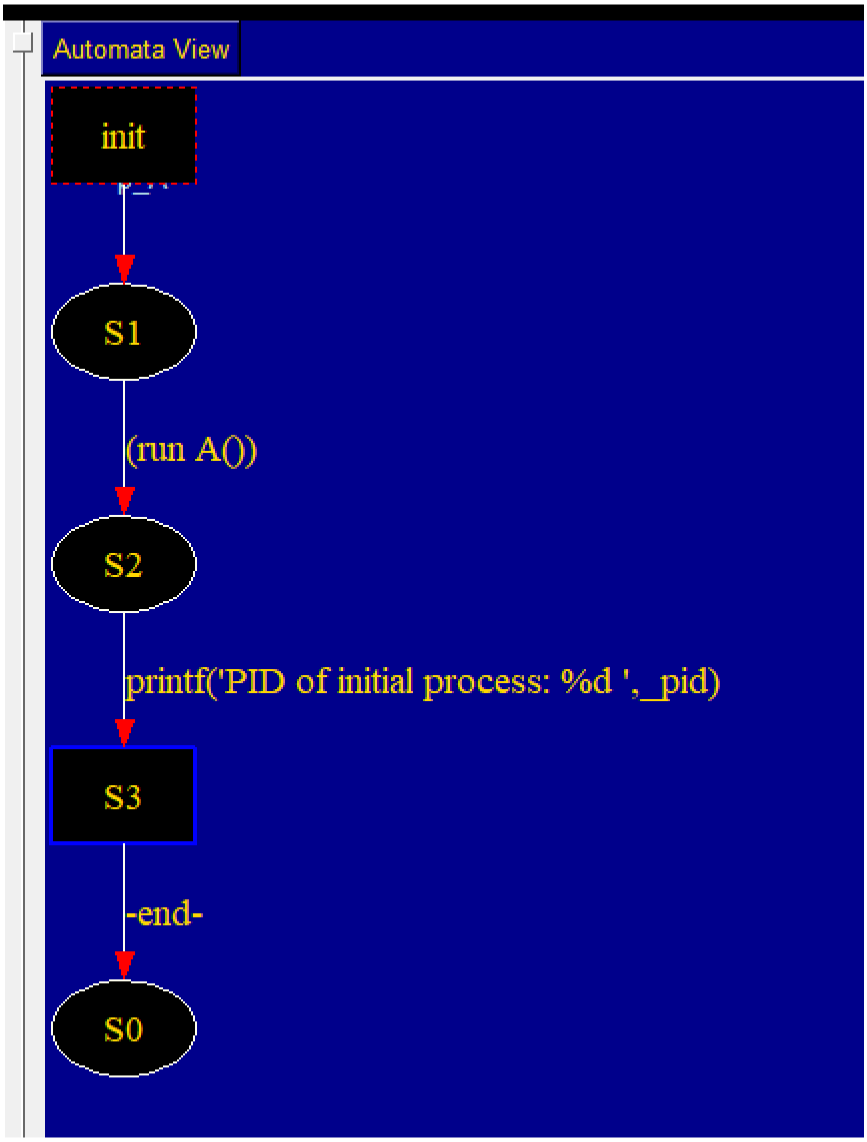
Q1:



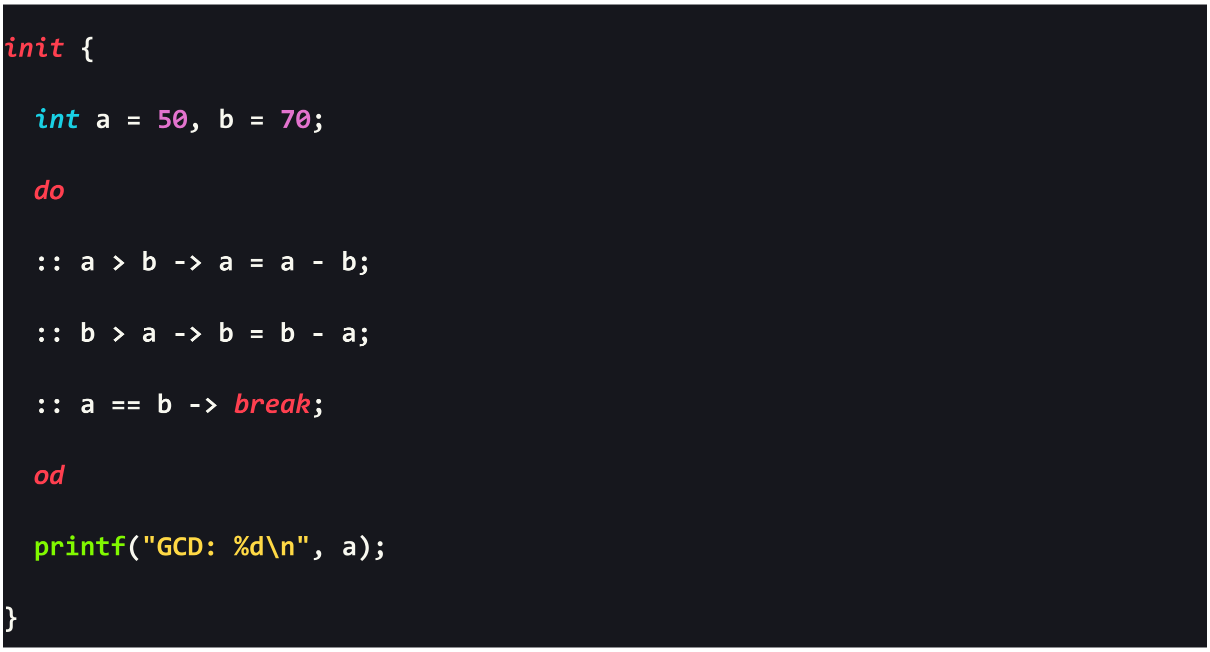
```
init {
  run A();
  printf("PID of initial process: %d\n", _pid);
}


proctype A() {
  printf("PID of child process: %d\n", _pid);
}
```



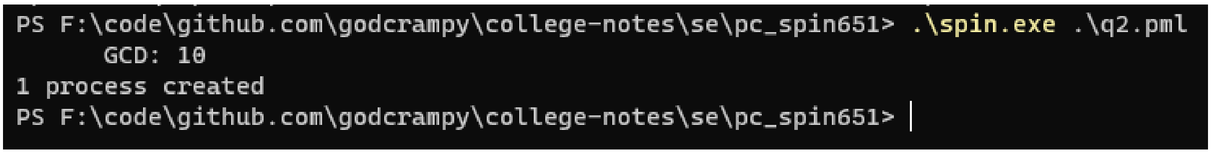
```
PS F:\code\github.com\godcrampy\college-notes\se\pc_spin651> .\spin.exe .\q1.pml
          PID of child process: 1
      PID of initial process: 0
2 processes created
PS F:\code\github.com\godcrampy\college-notes\se\pc_spin651> |
```


Automata View
init
S1
(run A())
S2
printf('PID of initial process: %d ',_pid)
S3
-end-
S0

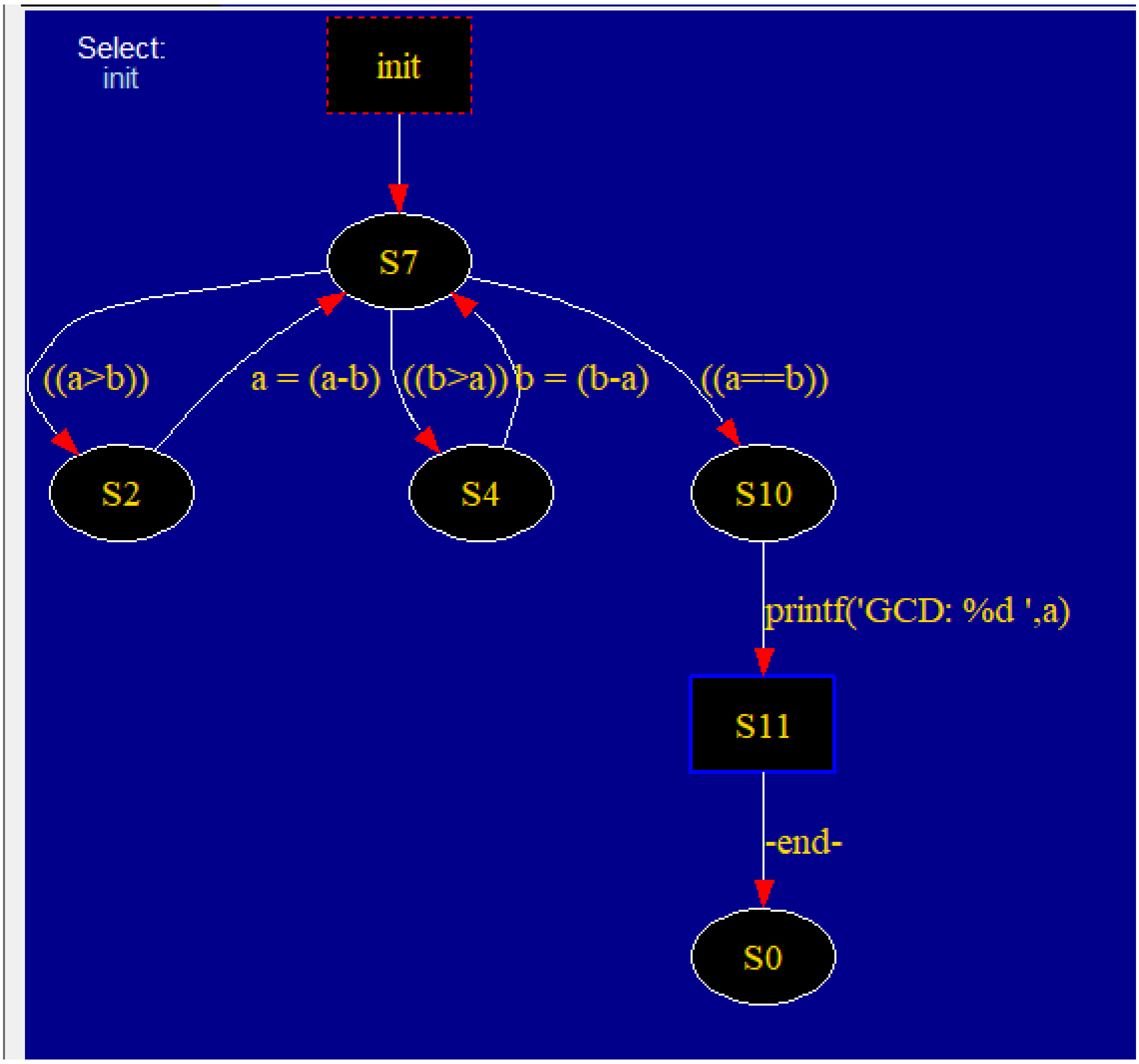
Q2:



```
init {

  int a = 50, b = 70;

  do

  :: a > b -> a = a - b;

  :: b > a -> b = b - a;

  :: a == b -> break;

  od

  printf("GCD: %d\n", a);

}
```

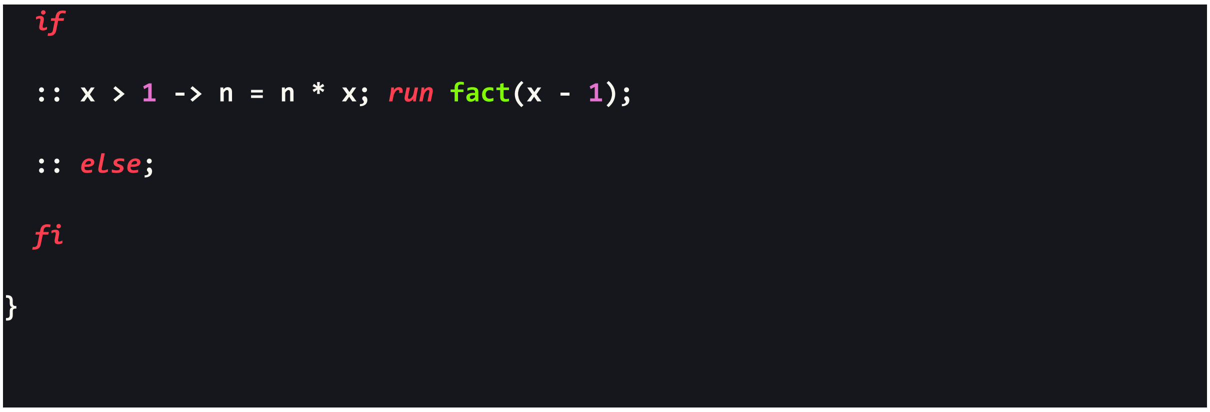


```
PS F:\code\github.com\godcrampy\college-notes\se\pc_spin651> .\spin.exe .\q2.pml
      GCD: 10
1 process created
PS F:\code\github.com\godcrampy\college-notes\se\pc_spin651> |
```

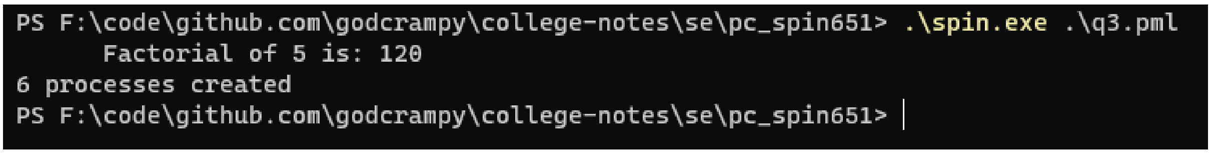


q3:

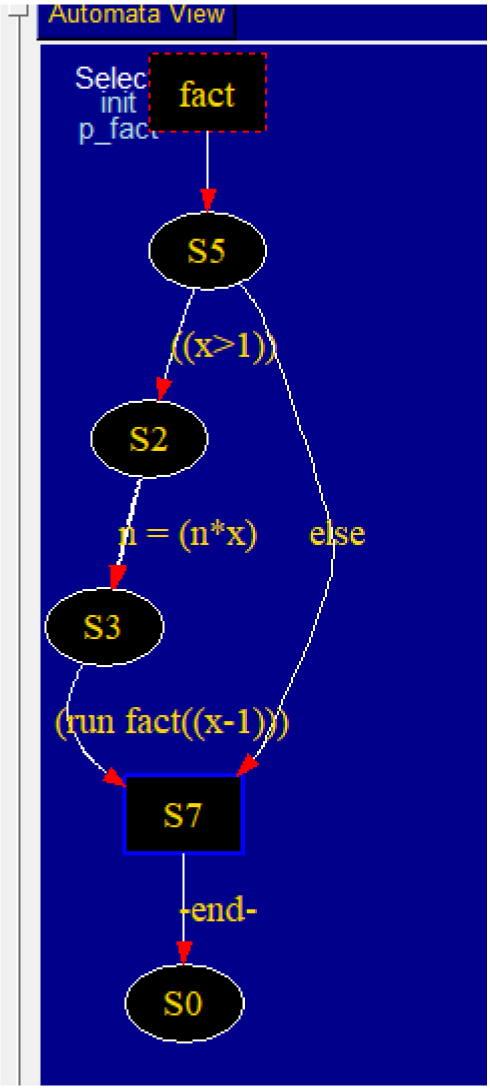
```
int n = 1;


init {
  int x = 5;
  run fact(x);
  (_nr_pr == 1) -> printf("Factorial of %d is: %d\n", x, n);
}


proctype fact(int x) {
```

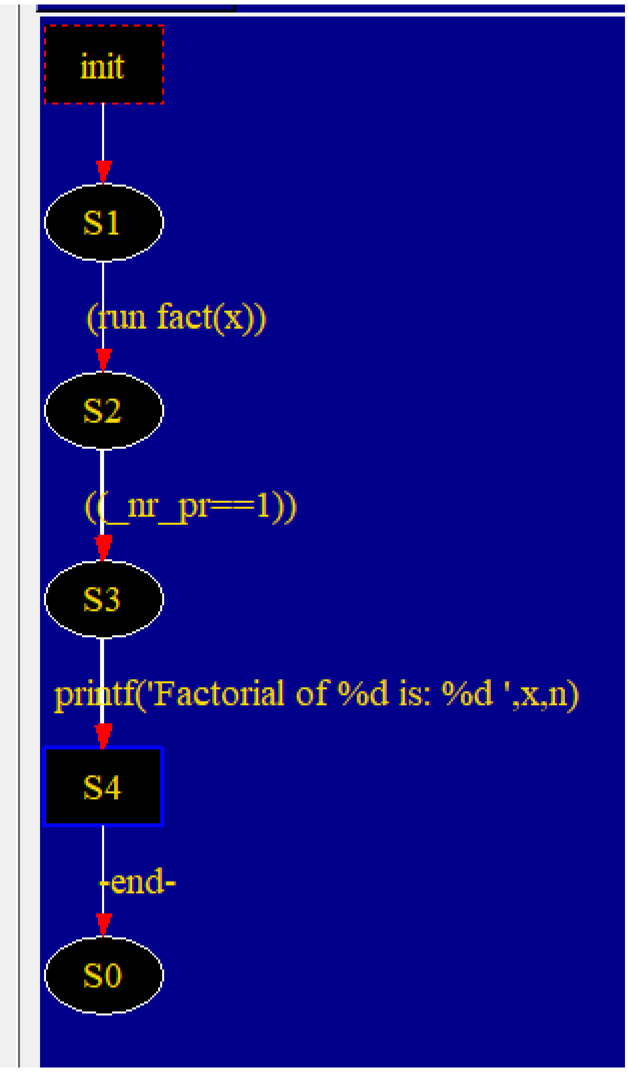


```
  if

  :: x > 1 -> n = n * x; run fact(x - 1);

  :: else;

  fi

}
```

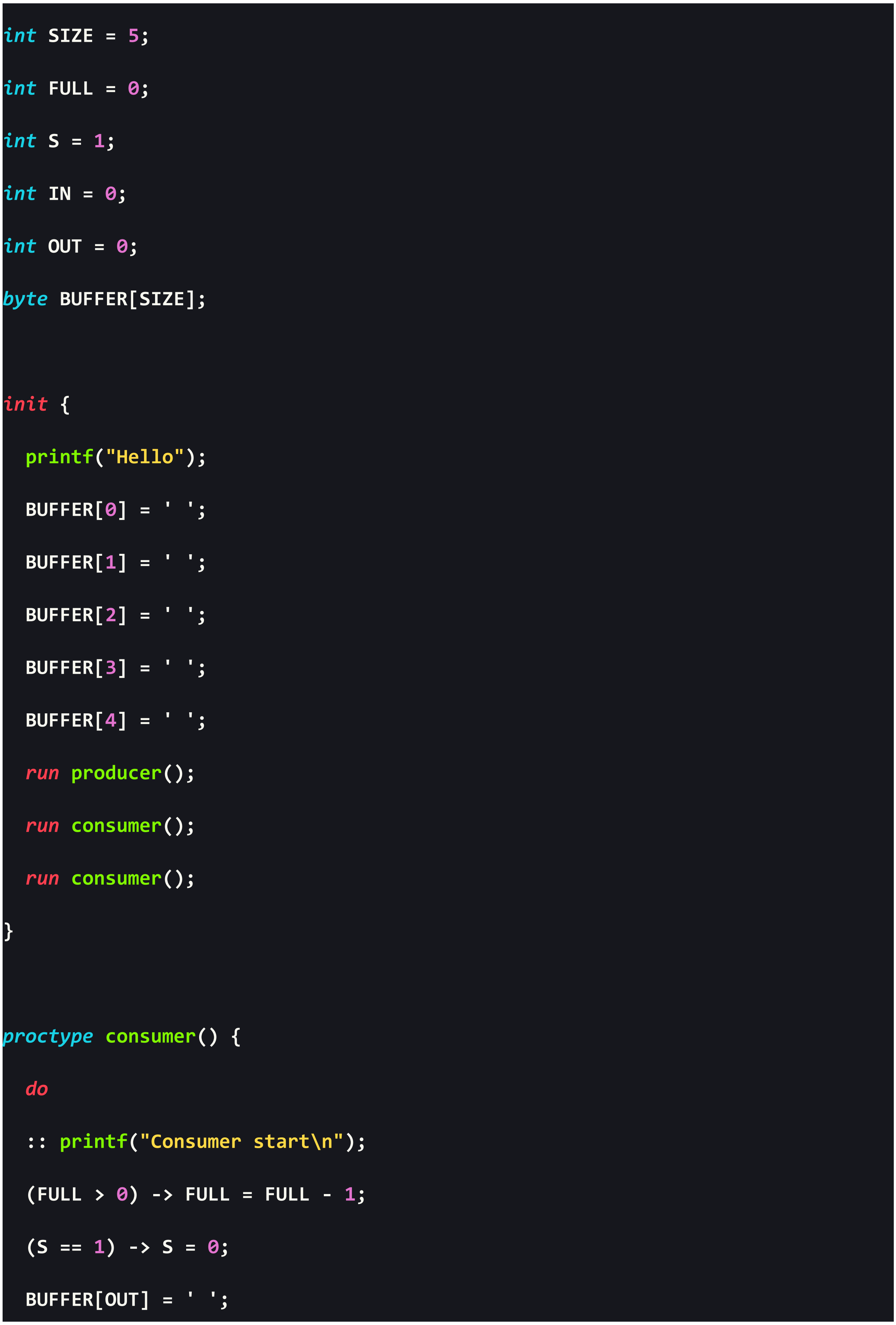


```
PS F:\code\github.com\godcrampy\college-notes\se\pc_spin651> .\spin.exe .\q3.pml
      Factorial of 5 is: 120
6 processes created
PS F:\code\github.com\godcrampy\college-notes\se\pc_spin651>
```

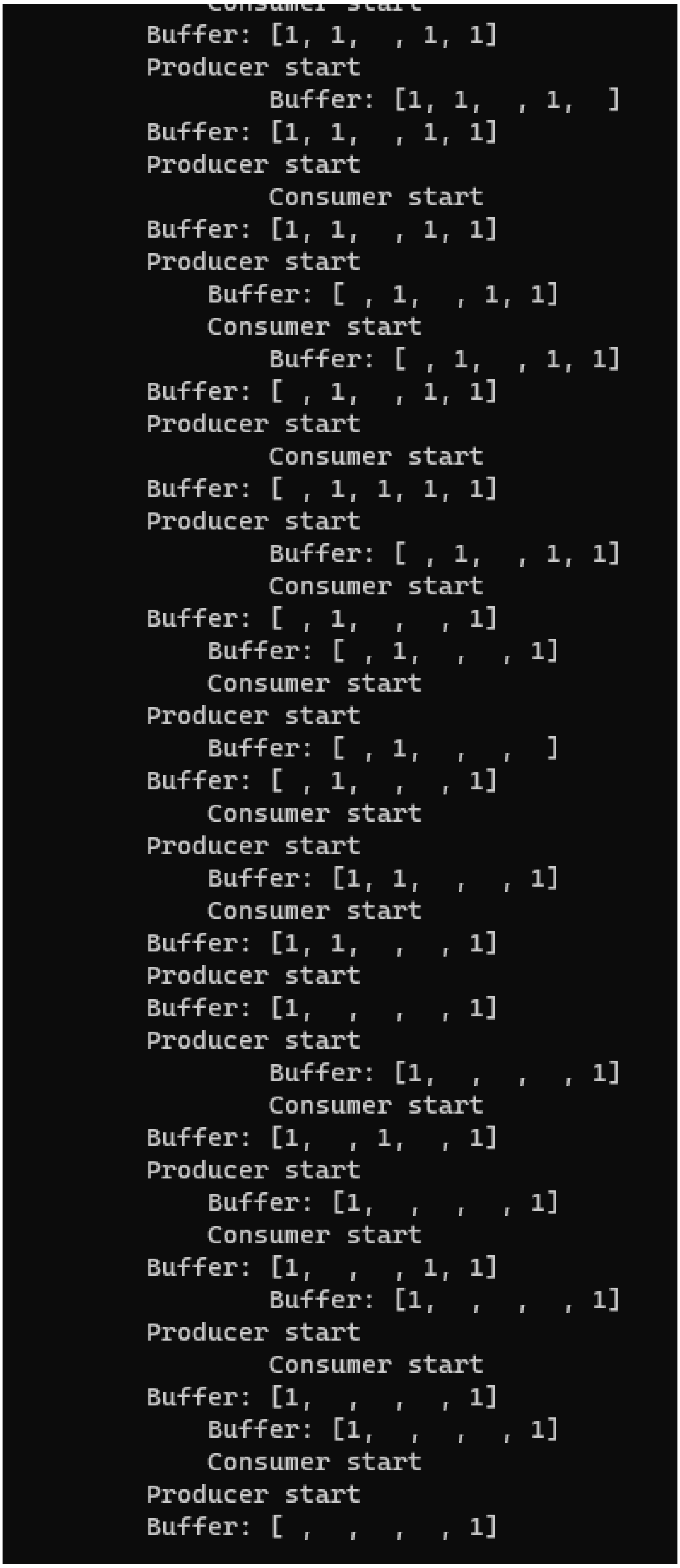




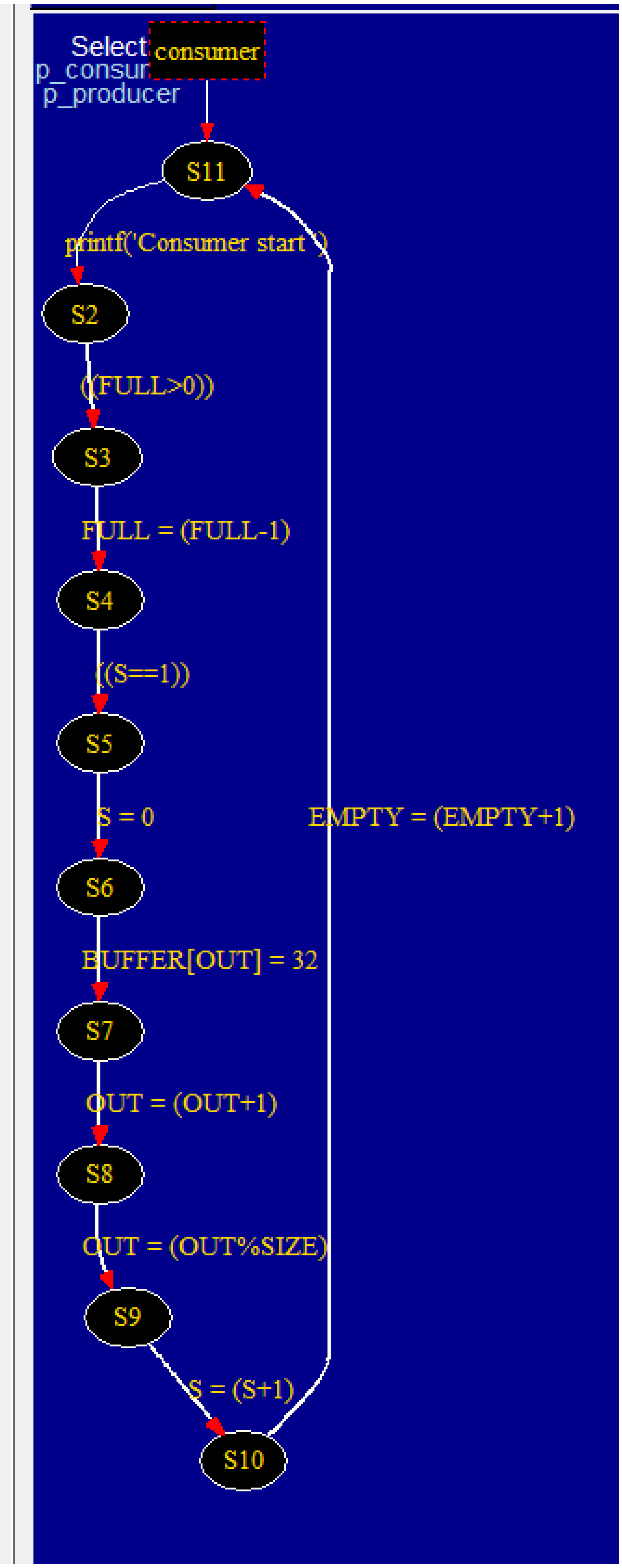
q4:

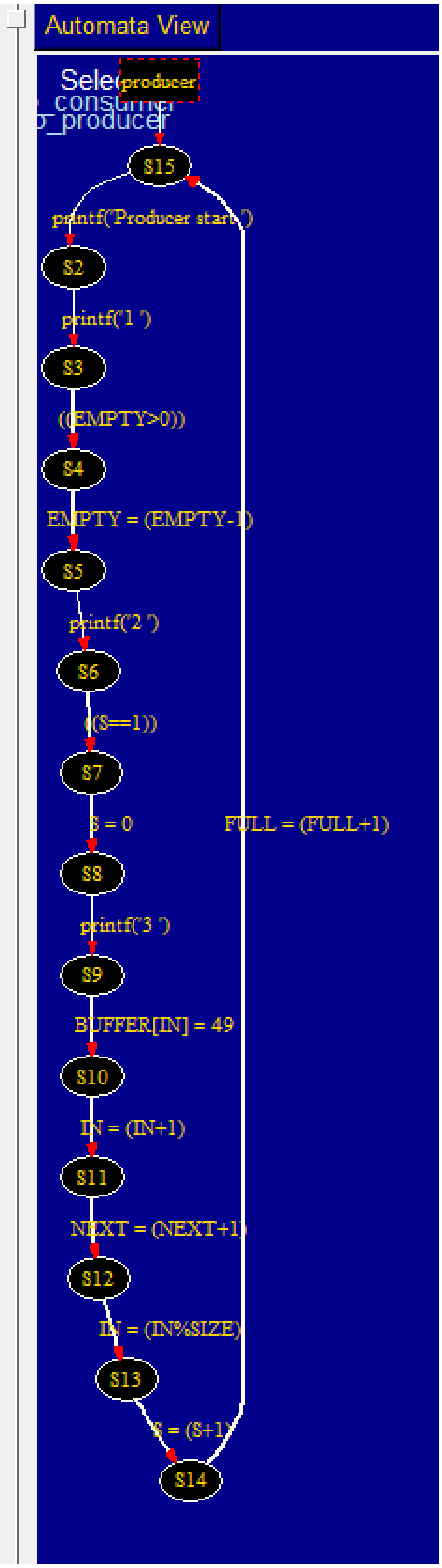


```
int SIZE = 5;

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

}
```



```
    Consumer start
Buffer: [1, 1,  , 1, 1]
Producer start
        Buffer: [1, 1,  , 1,  ]
Buffer: [1, 1,  , 1, 1]
Producer start
        Consumer start
Buffer: [1, 1,  , 1, 1]
Producer start
    Buffer: [ , 1,  , 1, 1]
    Consumer start
        Buffer: [ , 1,  , 1, 1]
Buffer: [ , 1,  , 1, 1]
Producer start
        Consumer start
Buffer: [ , 1, 1, 1, 1]
Producer start
        Buffer: [ , 1,  , 1, 1]
        Consumer start
Buffer: [ , 1,  ,  , 1]
    Buffer: [ , 1,  ,  , 1]
    Consumer start
Producer start
    Buffer: [ , 1,  ,  ,  ]
Buffer: [ , 1,  ,  , 1]
    Consumer start
Producer start
    Buffer: [1, 1,  ,  , 1]
    Consumer start
Buffer: [1, 1,  ,  , 1]
Producer start
Buffer: [1,  ,  ,  , 1]
Producer start
        Buffer: [1,  ,  ,  , 1]
        Consumer start
Buffer: [1,  , 1,  , 1]
Producer start
    Buffer: [1,  ,  ,  , 1]
    Consumer start
Buffer: [1,  ,  , 1, 1]
        Buffer: [1,  ,  ,  , 1]
Producer start
        Consumer start
Buffer: [1,  ,  ,  , 1]
    Buffer: [1,  ,  ,  , 1]
    Consumer start
Producer start
Buffer: [ ,  ,  ,  , 1]
```


Select
consumer
p_consur
p_producer
S11
printf('Consumer start ')
S2
((FULL>0))
S3
FULL = (FULL-1)
S4
((S==1))
S5
S = 0
EMPTY = (EMPTY+1)
S6
BUFFER[OUT] = 32
S7
OUT = (OUT+1)
S8
OUT = (OUT%SIZE)
S9
S = (S+1)
S10


Automata View
Selec
producer
consumer
p_producer
S15
printf('Producer start ')
S2
printf('1 ')
S3
((EMPTY>0))
S4
EMPTY = (EMPTY-1)
S5
printf('2 ')
S6
((S==1))
S7
S = 0
FULL = (FULL+1)
S8
printf('3 ')
S9
BUFFER[IN] = 49
S10
IN = (IN+1)
S11
NEXT = (NEXT+1)
S12
IN = (IN%SIZE)
S13
S = (S+1)
S14