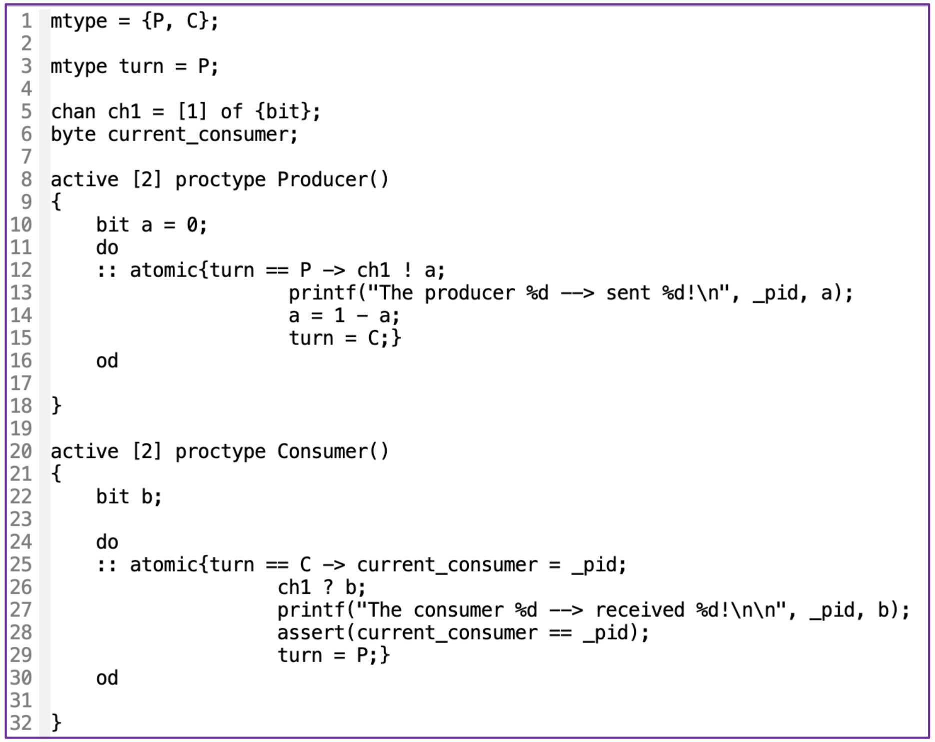
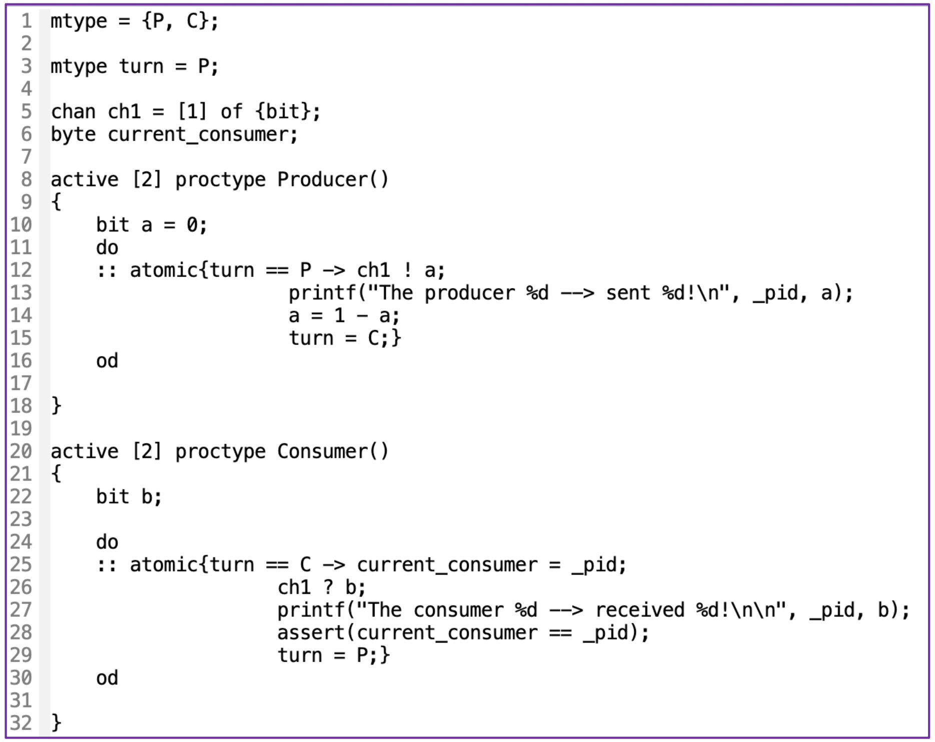
Q3: verify with LTL that once a value is produced, it will eventually be consumed. You can extend the model to support the verification if needed (Hints: use two global variables “produced” and “consumed”, to represent what has been produced and what has been consumed).



```
mtype = {P, C};

mtype turn = P;

chan ch1 = [1] of {bit};
byte current_consumer;

active [2] proctype Producer()
{
    bit a = 0;
    do
    :: atomic{turn == P -> ch1 ! a;
                     printf("The producer %d --> sent %d!\n", _pid, a);
                     a = 1 - a;
                     turn = C;}
    od

}

active [2] proctype Consumer()
{
    bit b;

    do
    :: atomic{turn == C -> current_consumer = _pid;
                    ch1 ? b;
                    printf("The consumer %d --> received %d!\n\n", _pid, b);
                    assert(current_consumer == _pid);
                    turn = P;}
    od

}
```

prod_cons_4.pml

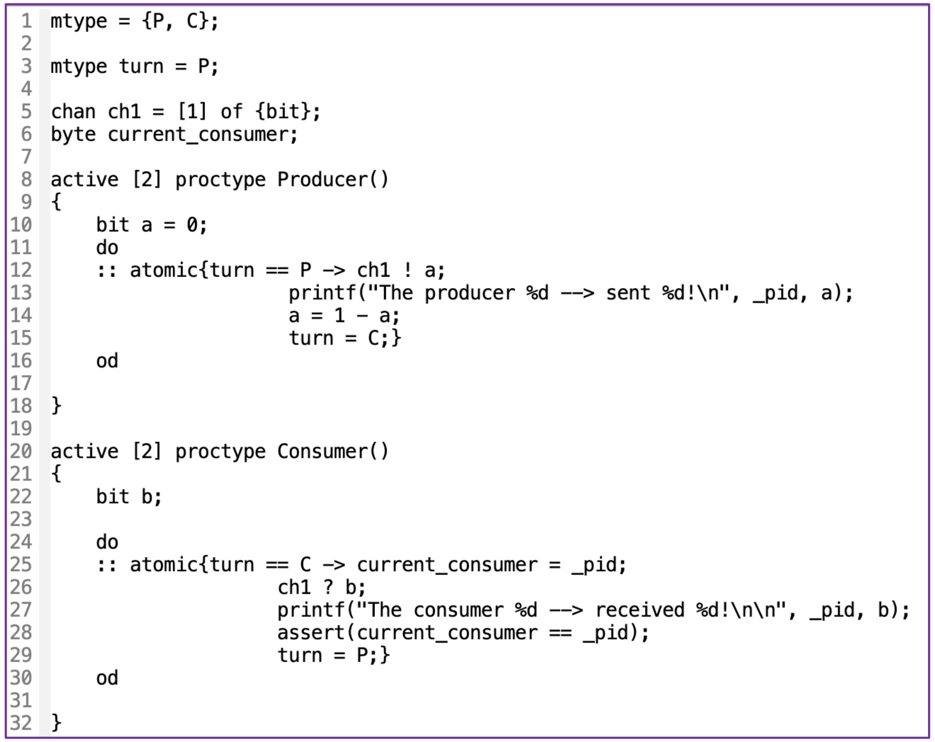
Q4: verify with LTL that it never happen that the same consumer consumes twice in a row. (Hints, use a global variable “previous_consumer”. Pay attention, global variables if not initialized, default value will be 0.)



```
mtype = {P, C};

mtype turn = P;

chan ch1 = [1] of {bit};
byte current_consumer;

active [2] proctype Producer()
{
    bit a = 0;
    do
    :: atomic{turn == P -> ch1 ! a;
                     printf("The producer %d --> sent %d!\n", _pid, a);
                     a = 1 - a;
                     turn = C;}
    od

}

active [2] proctype Consumer()
{
    bit b;

    do
    :: atomic{turn == C -> current_consumer = _pid;
                    ch1 ? b;
                    printf("The consumer %d --> received %d!\n\n", _pid, b);
                    assert(current_consumer == _pid);
                    turn = P;}
    od

}
```

prod_cons_4.pml

Q5: extend the model so that the same consumer does not consume twice in a row and verify it with LTL (Hints, use a global variable “previous_consumer”. Pay attention, global variables if not initialized, default value will be 0.)



```
mtype = {P, C};

mtype turn = P;

chan ch1 = [1] of {bit};
byte current_consumer;

active [2] proctype Producer()
{
    bit a = 0;
    do
    :: atomic{turn == P -> ch1 ! a;
                    printf("The producer %d --> sent %d!\n", _pid, a);
                    a = 1 - a;
                    turn = C;}
    od

}

active [2] proctype Consumer()
{
    bit b;

    do
    :: atomic{turn == C -> current_consumer = _pid;
                   ch1 ? b;
                   printf("The consumer %d --> received %d!\n\n", _pid, b);
                   assert(current_consumer == _pid);
                   turn = P;}
    od

}
```

prod_cons_4.pml

Q6: propose a new property relevant to the model, specify it in LTL and verify it with SPIN

Assignment: solve Q3 – Q6, submit a short report in PDF by Dec. 17th, 23:59. Late submission will NOT be accepted. For each question, you need to include:

1. the extended model with suitable explanation why it is extended as it is, if the model is extended;
2. the LTL property with suitable explanation why it is specified in the way it is;
3. answer whether the property is a safety property or a liveness property;
4. screenshot of the verification output, which must contain also the commands you used to run the verification;
5. If the property doesn't satisfy, run the guided simulation and explains the counterexample found.