First exercise for Proof Assistants 2009

The exercise is to write a Mizar formalization of the following mathematics:

> **Lemma 1.** Each natural number $n \equiv 3 \pmod 4$ has a prime divisor p for which it also holds that $p \equiv 3 \pmod 4$.
>
> *Proof.* We prove this by induction on the size of n. Take a prime divisor p of n. Because n is odd, p will be either 1 or $3 \pmod 4$. In the latter case we are finished, so suppose $p \equiv 1 \pmod 4$. Then n/p will be smaller than n and also be $3 \pmod 4$, The induction hypothesis then applies to n/p giving us a prime divisor that is $3 \pmod 4$ of n/p, but this will also be a prime divisor of n, giving us what we are looking for. □
>
> **Lemma 2.** For each natural number n there exists a prime number $p > n$ with $p \equiv 3 \pmod 4$.
>
> *Proof.* Using the previous lemma obtain a prime divisor $p \equiv 3 \pmod 4$ of $4n! - 1$. We show with a proof by contradiction that $p > n$. So suppose that $p \leq n$. Then p would divide $n!$, because it is one of the factors. But from that it follows that it also divides $4n!$, while by construction it already divided $4n! - 1$. Together this gives that it divides 1, which is impossible. □
>
> **Theorem.** The set $\{p \mid p \text{ is prime } \& \ p \equiv 3 \pmod 4\}$ is infinite.
>
> *Proof.* Immediate from Lemma 2. □

The exercise is to write a Mizar article that is error-free, and is in a shape that makes it fit to be submitted to the MML library.

The deadline for this exercise is Friday **June 12**. As a half-way checkpoint, Friday **April 24** a version of the article has to be handed in that already has the formalization of the statements of the two lemmas and the theorem, and that only has `*1` and `*4` errors left.

*

Some remarks about this exercise:

- I took me about two hours to do this exercise (but then I am an experienced Mizar user). My solution currently is 138 lines long.
- The proof of the second lemma is an adapted version of Euclid's proof that there are infinitely many primes, which goes like this:

  > **Lemma 2^-.** For each natural number n there exists a prime number $p > n$.
  >
  > *Proof.* Consider a prime divisor p of $n! + 1$. We show with a proof by contradiction that $p > n$. So suppose that $p \leq n$. Then p would

divide $n!$, because it is one of the factors. But by construction it already divided $n! + 1$. Together this gives that it divides 1, which is impossible. □

Here is a Mizar formalization of this simpler proof (which is 30 lines long):

```
environ
  vocabularies FILTER_0, QC_LANG1, ARYTM_3, ORDINAL2, ARYTM;
  notations NAT_1, INT_2, XXREAL_0, ORDINAL1, NUMBERS,
    XCMPLX_0, NEWTON, SUBSET_1, INT_1;
  constructors NAT_1, XXREAL_0, NEWTON, NAT_D;
  registrations XXREAL_0, ORDINAL1, INT_1;
  requirements SUBSET, BOOLE, NUMERALS, REAL, ARITHM;
  theorems INT_2, NAT_1, NEWTON, XREAL_1, WSIERP_1;

begin
  reserve n for natural number;
  reserve p for Element of NAT;

theorem
  for n ex p st p is prime & p > n
proof
  let n;
  n! > 0 by NEWTON:23;
  then n! >= 0 + 1 by NAT_1:13;
  then n! + 1 >= 1 + 1 by XREAL_1:8;
  then consider p such that
A1: p is prime & p divides n! + 1 by INT_2:48;
A2: p <> 0 & p > 1 by A1,INT_2:def 5;
  take p;
  thus p is prime by A1;
  assume p <= n;
  then p divides n! by A2,NEWTON:54;
  hence contradiction by A1,A2,INT_2:1,WSIERP_1:20;
end;
```

– If you think of a useful lemma, first try to find it in the MML. If you cannot find it, do not hesitate to state and prove it yourself (but only prove it *after* you finished the main proof.) For example, the lemma

```
for n,a,b being Integer holds
  (a * b) mod n = ((a mod n) * (b mod n)) mod n
```

is in the MML (find it!) However, the lemma

```
n is even iff n mod 4 = 0 or n mod 4 = 2
```

does not seem to be there yet. In my formalization the proof of this lemma took 23 lines.

– Be aware of the difference between

```
natural number
```

and

```
Element of NAT
```

To get from the first to the second, use `reconsider` with `ORDINAL1:def 13`. It is crazy that this is needed, but each time you introduce a variable you need to decide which of these two types is the most appropriate for that specific variable.

- A `scheme` that you can use for the induction in the proof of the first lemma is `NAT_1:sch 4`.
- Finally, here is – for a random example – an easy way to 'calculate' the value of a '`mod`' expression:

```
42 = 4*10 + 2 & 2 < 4;
then 42 mod 4 = 2 by NAT_D:def 2;
```