```
theory Warmup-Problem-A
  imports
    Complex-Main
    HOL−Number-Theory.Cong
begin
```

0.1 Warmup problem A

Solve the equation $3^x = 4y + 5$ in the integers.

We begin with the following lemma:

```
lemma even-power-3: [3^k = 1::int] (mod 4) ⟷ even k
proof −
  have [3^k = (−1::int)^k] (mod 4)
    by (intro cong-pow) (auto simp: cong-def)
  thus ?thesis
    by (auto simp: cong-def minus-one-power-iff)
qed
```

Here is an alternative proof — hopefully it will be instructive in doing calculations mod n.

```
lemma [3^k = 1::int] (mod 4) ⟷ even k
proof (cases even k)
  case True
  then obtain l where 2∗l = k by auto
  then have [3^k = (3^2)^l] (mod 4) (is cong - ... -)
    by (auto simp add: power-mult)
  also have [... = (1::int)^l] (mod 4) (is cong - ... -)
    by (intro cong-pow) (simp add: cong-def)
  also have [... = 1] (mod 4) by auto
  finally have [3^k = 1::int] (mod 4).
  thus ?thesis using ‹even k› by blast
next
  case False
  then obtain l where 2∗l + 1 = k
    using oddE by blast
  then have [3^k = 3^(2∗l+1)] (mod 4) (is cong - ... -) by auto
  also have [... = (3^2)^l ∗ 3] (mod 4) (is cong - ... -)
    by (metis power-mult power-add power-one-right cong-def)
  also have [... = (1::int)^l ∗ 3] (mod 4) (is cong - ... -)
    by (intro cong-mult cong-pow) (auto simp add: cong-def)
  also have [... = 3] (mod 4) by auto
  finally have [3^k ≠ 1::int] (mod 4) by (auto simp add: cong-def)
  then show ?thesis using ‹odd k› by blast
qed
```

This allows us to prove the theorem, provided we assume x is a natural number.

```
theorem warmupA-natx:
  fixes x :: nat and y :: int
  shows 3^x = 4∗y + 5 ⟷ even x ∧ y = (3^x − 5) div 4
proof −
  have even x ∧ y = (3^x − 5) div 4 if 3^x = 4∗y + 5
  proof −
    from that have [3^x = 4∗y + 5] (mod 4) by auto
    also have [4∗y + 5 = 5] (mod 4)
      by (metis cong-mult-self-left cong-add-rcancel-0)
    also have [5 = 1::int] (mod 4) by (auto simp add: cong-def)
    finally have [(3::int)^x = 1] (mod 4).
    hence even x using even-power-3 by auto
    thus ?thesis using that by auto
  qed
  moreover have 3 ^ x = 4 ∗ y + 5 if even x ∧ y = (3^x − 5) div 4
```

```
  proof -
    from that have even x and y-form: y = (3^x - 5) div 4 by auto
    then have [3^x = 1::int] (mod 4) using even-power-3 by blast
    then have ((3::int)^x - 5) mod 4 = 0
      by (simp add: cong-def mod-diff-cong)
    thus ?thesis using y-form by auto
  qed
  ultimately show ?thesis by blast
qed
```

To consider negative values of x, we'll need to venture into the reals:

```
lemma powr-int-pos:
  fixes x y :: int
  assumes *: 3 powr x = y
  shows x ≥ 0
proof (rule ccontr)
  assume neg-x: ¬ x ≥ 0
  then have y-inv: y = inverse ((3::nat)^nat (-x)) (is y = inverse (?n::nat))
    using powr-real-of-int and * by auto
  hence real ?n * of-int y = 1 by auto
  hence ?n * y = 1 using of-int-eq-iff by fastforce
  hence ?n = 1
   by (metis nat-1-eq-mult-iff nat-int nat-numeral-as-int numeral-One of-nat-mult zmult-eq-1-iff)

  hence nat (-x) = 0 by auto
  thus False using neg-x by auto
qed

corollary warmupA:
  fixes x y :: int
  shows 3 powr x = 4*y + 5 ⟷ x ≥ 0 ∧ even x ∧ y = (3^(nat x) - 5) div 4
proof
  assume assm: 3 powr x = 4*y + 5
  then have x ≥ 0 using powr-int-pos by fastforce
  hence 3 powr (nat x) = 4*y + 5 using assm by simp
  hence (3::real)^(nat x) = 4*y + 5 using powr-realpow by auto
  hence with-nat: 3^(nat x) = 4*y + 5 using of-int-eq-iff by fastforce
  hence even (nat x) ∧ y = (3^(nat x) - 5) div 4 using warmupA-natx by auto
  thus x ≥ 0 ∧ even x ∧ y = (3^(nat x) - 5) div 4 using ‹x ≥ 0› and even-nat-iff by auto
next
  assume assm: x ≥ 0 ∧ even x ∧ y = (3^(nat x) - 5) div 4
  then have 3^(nat x) = 4*y + 5 using warmupA-natx and even-nat-iff by blast
  thus 3 powr x = 4*y + 5 using assm powr-real-of-int by fastforce
qed

end
```