Solution 1

Step one.

Create a procedure find-roots. There are several ways, but my favorite is, using the theorem below as the basis for the algorithm:

In algebra, the rational root theorem (or rational root test, rational zero theorem, rational zero test or p/q theorem) states a constraint on rational solutions of a polynomial equation

\[ a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = 0 \]

with integer coefficients. These solutions are the possible roots (equivalently, zeroes) of the polynomial on the left side of the equation.

If \( a_0 \) and \( a_n \) are nonzero, then each rational solution \( x \), when written as a fraction \( x = p/q \) in lowest terms (i.e., the greatest common divisor of \( p \) and \( q \) is 1), satisfies

- \( p \) is an integer factor of the constant term \( a_0 \), and
- \( q \) is an integer factor of the leading coefficient \( a_n \).

The rational root theorem is a special case (for a single linear factor) of Gauss's lemma on the factorization of polynomials. The integral root theorem is a special case of the rational root theorem if the leading coefficient \( a_n = 1 \).

Step 2.

Use Turing reduction, to get a special case of Rice's theorem.

Recall we encode a polynomial by (list n coeff) which I shall call the argument "poly".
(define Turing-reduce
  (lambda ((p <procedure>) (a <object>))
    (list (lambda (poly)
      (p a)
      (find-roots poly))))))

(define safe?
  (lambda ((p <procedure>) (a <object>))
    (apply poly-check (Turing-reduce p a))))

We would also accept:

(define Safe?
  (lambda ((p <procedure>) (a <object>))
    (Poly-check (lambda (poly)
      (p a)
      (find-roots poly))))))

Or, some students might write:

(define Safe?
  (lambda ((p <procedure>) (a <object>))
    (let ((f <procedure>)
      (lambda (poly)
        (p a)
        (find-roots poly)))
      (Poly-check f)))

In all cases:
NOS the reduction satisfies the IFF condition in the definition of Turing reduction.

**Solution 2.**

reduce Equiv? to Poly-check. You must define *find-roots* first. Make sure it is a provable Turing reduction, and make sure it goes the right way