

```
/*
 * transcendentals.c
 *
 * What is the value of acos(1.0000000001)?
 * On the Mac you get the intended answer (0.0), but unix returns NaN.
 * To guard against this sort of problem, the SnapPea kernel uses
 *
 *     double safe_acos(double x);
 *     double safe_asin(double x);
 *     double safe_sqrt(double x);
 *
 * Incredibly enough, the standard ANSI libraries omit the
 * inverse hyperbolic trig functions entirely.
 *
 *     double arcsinh(double x);
 *     double arccosh(double x);
 *
 * Many (but not all) standard libraries now provide asinh() and acosh().
 * I've changed the names of my homemade versions to arcsinh() and arccosh()
 * to avoid conflicts. 2000/02/20 JRW
 */

#include "kernel.h"

#define ERROR_EPSILON 1e-3

double safe_acos(double x)
{
    if (x > 1.0)
    {
        if (x > 1.0 + ERROR_EPSILON)
            uFatalError("safe_acos", "transcendentals");
        x = 1.0;
    }
    if (x < -1.0)
    {
        if (x < -(1.0 + ERROR_EPSILON))
            uFatalError("safe_acos", "transcendentals");
        x = -1.0;
    }

    return acos(x);
}

double safe_asin(double x)
{
    if (x > 1.0)
    {
        if (x > 1.0 + ERROR_EPSILON)
            uFatalError("safe_asin", "transcendentals");
        x = 1.0;
    }
    if (x < -1.0)
    {
        if (x < -(1.0 + ERROR_EPSILON))
            uFatalError("safe_asin", "transcendentals");
        x = -1.0;
    }

    return asin(x);
}

double safe_sqrt(double x)
{
    if (x < 0.0)
    {
        if (x < -ERROR_EPSILON)
            uFatalError("safe_sqrt", "transcendentals");
        x = 0.0;
    }
}
```

```
    return sqrt(x);
}

double arcsinh(
    double x)
{
    return log(x + sqrt(x*x + 1.0));
}

double arccosh(
    double x)
{
    if (x < 1.0)
    {
        if (x < 1.0 - ERROR_EPSILON)
            uFatalError("arccosh", "transcendentals");
        x = 1.0;
    }

    return log(x + sqrt(x*x - 1.0));
}
```