```
import numpy as np
from operator import add, gt
from core import *

def test_binary_function():
    expr = "(def (fun x y) (add x y))"
    fun = get_function(expr, "fun")
    assert fun(4, 3) == 7

def test_multiline_function():
    expr = "(def (fun x y) (def a (add y x)) (add a 1))"
    fun = get_function(expr, "fun")
    assert fun(4, 3) == 8

def test_multiple_function():
    expr = """(def (fun1 x y) (add x y))
              (def (fun2 x) (add x x))"""
    fun1 = get_function(expr, "fun1")
    fun2 = get_function(expr, "fun2")

    assert fun1(4, 3) == 7
    assert fun2(5) == 10

def test_if_else():
    expr = "(def (fun x y) (if (gt x y) x y))"
    fun = get_function(expr, "fun")
    assert fun(4, 3) == 4
    assert fun(8, 10) == 10

def test_grad_add():
    expr = """(def (f x y) (pow x y))
              (def df (grad f 0))"""
    df = get_function(expr, "df")
```

```
        assert np.allclose(df(3.0, 4), 108.0)

    def test_grad_sin():
        expr = """(def (f x) (np.sin x))
                  (def df (grad f 0))"""
        df = get_function(expr, "df")
        assert np.allclose(df(np.pi/3), 0.5)

    def test_grad_fanout():
        expr = """(def (f x) (add (np.sin x) (np.sin x)))
                  (def df (grad f 0))"""
        df = get_function(expr, "df")
        assert np.allclose(df(np.pi/3), 1.0)

    def test_grad_const():
        expr = """(def (f x) 1)
                  (def df (grad f 0))"""
        df = get_function(expr, "df")
        assert np.allclose(df(2.0), 0.0)

    def test_grad_exp():
        expr = """(def (f x) (np.exp x))
                  (def df (grad f 0))"""
        df = get_function(expr, "df")
        assert np.allclose(df(2.0), np.exp(2.0))

    def test_double_grad_exp():
        expr = """(def (f x) (np.exp x))
                  (def df (grad f 0))
                  (def ddf (grad df 0))"""
        ddf = get_function(expr, "ddf")
        assert np.allclose(ddf(2.0), np.exp(2.0))

    def test_grad_identity():
        expr = """(def (f x) x)
                  (def df (grad f 0))"""
        df = get_function(expr, "df")
        assert np.allclose(df(2.0), 1.0)

    def test_double_grad_identity():
        expr = """(def (f x) x)
                  (def df (grad f 0))
                  (def ddf (grad df 0))"""
        ddf = get_function(expr, "ddf")
        print ddf(2.0)
        assert np.allclose(ddf(2.0), 0.0)

    def test_double_grad_sin():
        expr = """(def (f x) (np.sin x))
                  (def df (grad f 0))
                  (def ddf (grad df 0))"""
        ddf = get_function(expr, "ddf")
        print ddf(np.pi/6)
```

```
86         print ddf(np.pi/6)
87         assert np.allclose(ddf(np.pi/6), -0.5)
```