

```
1 import numpy as np
2 from operator import add, gt
3 from core import *
4
5 def test_binary_function():
6     expr = "(def (fun x y) (add x y))"
7     fun = get_function(expr, "fun")
8     assert fun(4, 3) == 7
9
10 def test_multiline_function():
11     expr = "(def (fun x y) (def a (add y x)) (add a 1))"
12     fun = get_function(expr, "fun")
13     assert fun(4, 3) == 8
14
15 ✓ def test_multiple_function():
16     expr = """"(def (fun1 x y) (add x y))
17             (def (fun2 x) (add x x))""""
18     fun1 = get_function(expr, "fun1")
19     fun2 = get_function(expr, "fun2")
20
21     assert fun1(4, 3) == 7
22     assert fun2(5) == 10
23
24 ✓ def test_if_else():
25     expr = "(def (fun x y) (if (gt x y) x y))"
26     fun = get_function(expr, "fun")
27     assert fun(4, 3) == 4
28     assert fun(8, 10) == 10
29
30 ✓ def test_grad_add():
31     expr = """"(def (f x y) (pow x y))
32             (def df (grad f 0))""""
33     df = get_function(expr, "df")
```

```

34         assert np.allclose(df(3.0, 4), 108.0)
35
36 ✓ def test_grad_sin():
37     expr = """(def (f x) (np.sin x))
38             (def df (grad f 0))"""
39     df = get_function(expr, "df")
40     assert np.allclose(df(np.pi/3), 0.5)
41
42 ✓ def test_grad_fanout():
43     expr = """(def (f x) (add (np.sin x) (np.sin x)))
44             (def df (grad f 0))"""
45     df = get_function(expr, "df")
46     assert np.allclose(df(np.pi/3), 1.0)
47
48 ✓ def test_grad_const():
49     expr = """(def (f x) 1)
50             (def df (grad f 0))"""
51     df = get_function(expr, "df")
52     assert np.allclose(df(2.0), 0.0)
53
54 ✓ def test_grad_exp():
55     expr = """(def (f x) (np.exp x))
56             (def df (grad f 0))"""
57     df = get_function(expr, "df")
58     assert np.allclose(df(2.0), np.exp(2.0))
59
60 ✓ def test_double_grad_exp():
61     expr = """(def (f x) (np.exp x))
62             (def df (grad f 0))
63             (def ddf (grad df 0))"""
64     ddf = get_function(expr, "ddf")
65     assert np.allclose(ddf(2.0), np.exp(2.0))
66
67 ✓ def test_grad_identity():
68     expr = """(def (f x) x)
69             (def df (grad f 0))"""
70     df = get_function(expr, "df")
71     assert np.allclose(df(2.0), 1.0)
72
73 ✓ def test_double_grad_identity():
74     expr = """(def (f x) x)
75             (def df (grad f 0))
76             (def ddf (grad df 0))"""
77     ddf = get_function(expr, "ddf")
78     print ddf(2.0)
79     assert np.allclose(ddf(2.0), 0.0)
80
81 ✓ def test_double_grad_sin():
82     expr = """(def (f x) (np.sin x))
83             (def df (grad f 0))
84             (def ddf (grad df 0))"""
85     ddf = get_function(expr, "ddf")
86     print ddf(np.pi/6)

```

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

---