WorkSheet: Testing 1

For each of the unit tests given below, state whether the test passes or fails. For each test that fails, explain why.

Problem 1.

```
def test_eq(a,b):
    assert (a == b)

test_eq(41,52)
```

Problem 2.

```
def test_list_eq():
    assert [0,1,2] == [0,1,3]

test_list_eq()
```

Problem 3.

```
def test_text():
     assert 'eggs' == 'eggs'

test_text()
```

Problem 4.

```
def test_dict_eq():
   assert {1:0,2:1}=={1:0,2:1}

test_dict_eq():
```

Problem 5.

```
def test_tuple(a,b):
    assert (b,a) == (1,2)

test_tuple(2,1)
```

Problem 6.

```
def test_nums():
    assert num1 * 2 < num2

num1 = 3
num2 = 6
test_nums()
```

Problem 7.

```
class Cat():
    a = 1

def test_attr():
    i = Cat()
    assert i.a == 2

test_attr()
```

Problem 8.

```
class Cat():
    a = 1
    def __init__(self):
        self.a=2

def test_Cat_att():
    i = Cat()
    assert i.a == 2

test_Cat_att()
```

Problem 9.

```python
is_cat = True
is_dog = False
def test_bool():
    assert is_cat or is_dog

test_bool()
```

Problem 10.

```python
def test_in_sentence(x):
    text= 'a␣cat␣is␣not␣a␣dog'
    assert x not in text

test_in_sentence('cats')
```

Problem 11.

```python
def test_set_compare(s1,s2):
    s1 = set("1234")
    s2 = set("2345")
    assert s1.issubset(s2)

s1=set("1234")
s2=set("12345")
test_set_compared(s1,s2)
```

Problem 12.

```python
def func():
    return False

def test_func():
    assert func() is False

test_func()
```

Problem 13.

```python
class Animal():
   b = 1

class Bug():
   b = 2

def test_b():
   assert Animal.b==Bug.b

test_b()
```

Problem 14.

```python
class MyClass:
    def__init__(self, max):
        self.max = max

def test_class():
    c1 = MyClass(20)
    c2 = MyClass(20)
    assert c1 == c2

test_class()
```

Problem 15.

```python
class Count():
    a=10
    def __init__(self):
        a=11

def test_Count():
    c = Count()
    assert c.a==11

test_Count()
```

Problem 16.

```python
def test_startswith():
    assert x().startswith(y())

def x():
    return "0101"

def y():
    return "10"

test_startswith()
```

Problem 17.

```
def test_expression():
    x = 6*7
    assert x == 42/2*4-42

test_expression()
```

Problem 18.

```
def test_expression2():
    x = 9*3/10+4.3
    assert x > 7

test_expression2()
```

Problem 19.

```
def test_list():
    alst = [1, 2, 3, 4, 5, 6]
    assert alst[-1] > alst[3]

test_list()
```

Problem 20.

```
def test_list2():
    alst = [1,[2],[3,4,5],6]
    assert alst[-1] > alst[3]

test_list2()
```