Assignment – Testing using Python

Ques 1) Write a Function for following --

- For a given list of 'n' integers, the function returns the index of the element with the minimum value in the list. If there is more than one element with the minimum value, the returned index should be the smallest one.
- If an empty list is passed to the function, it should raise an Exception.
- Design 4 separate unit test methods, testing if the function behaves correctly. (use unittest)

Ques 2) We have below code snippet saved in a file Calc.py –

```
def add(a,b):
    return a+b

def multiply (a,b):
    return a**b

def divide (a,b):
    return a//b
```

Write unit test cases to identify 3 issues with the functions above and fix them.

Ques 3) Given below code snippet –

```
class Employee:

    raise_money = 1.10
    salutation = ['Mr','Ms']

    def __init__(self, first, last, pay):
        self.first = first
        self.last = last
        self.pay = pay

    @property
    def email(self):
        return '{}.{}@email.com'.format(self.first, self.last)

    @property
    def fullname(self):
        return '{} {}'.format(self.first, self.last)

    def hike_salary(self):
        self.pay = int(self.pay * self.raise_money)
```

Write a class for unit test for following use stories –

a) Email details should be changing with the corresponding change in first and last name.
b) Full name of employee is being assigned properly when first and last names are changing.
c) Salary hike should be 10%.
d) Write 'setUp' and 'tearDown' methods to make your unittest reusable.

Ques 4) Write a function to calculate the sum of areas of two circles with radii 'r1' and 'r2' entered by the user. Prepare test methods for following user stories –

a) Area is computed properly
b) No negative values in the input
c) Input values should be real numbers

Ques 5) Complete below code to test for various strings functions –

```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        """ write your code here"""

    def test_isupper(self):
        """ write your code here """

    def test_split(self):
        s = 'hello world'

        """ write your code here to check that s.split fails when the
        separator is not a string """


if __name__ == '__main__':
    unittest.main()
```

Ques 6) Complete below code to test for various strings functions –

```
import unittest

class TestStringMethods(unittest.TestCase):

    def setUp(self):
        pass


    def test_strings_a(self):
        """ Returns True if the string contains 4 a """


    def test_upper(self):
        """ Returns True if the string is in upper case """


    def test_isupper(self):
        """Returns TRUE if the string is in uppercase
        else returns False """


    def test_strip(self):
        s = 'geeksforgeeks'
        """Returns true if the string is stripped and
        matches the given output """


    def test_split(self):
        s = 'hello world'
        """Returns true if the string splits and matches
         the given output """

if __name__ == '__main__':
    unittest.main()
```

Ques 7) Find problems in the following code –

```python
import doctest

def fib(n):
    """ Calculates the n-th Fibonacci number iteratively

    fib(0)
    0
    fib(1)
    1
    fib(10)
    55
    fib(40)
    102334155


    """
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)

if __name__ == "__main__":
    doctest.testmod()
```

Ques 8) What is the output of the following code snippet –

```python
f0 = open("foo.txt", "rw+")
print("Name of the file -- ", f0.name)

# Assuming file has following 5 lines
# This is the 1st line of the file
# This is the 2nd line of the file
# This is the 3rd line of the file
# This is the 4th line of the file
# This is the 5th line of the file

for index in range(5):
    line = f0.next()
    print("Line No {} ? {}".format(index, line))
    f0.close()
```

Ques 9) What is the output of the following code snippet –

```python
class A:
    def one(self):
        return self.two()

    def two(self):
        return 'A'

class B(A):
    def two(self):
        return B

obj1 = A()
obj2 = B()
print(obj1.two(), obj2.two())
```