IT 212 – Applied OO Programming

June 6, 2018 Name ________________________

Part A. Multiple Choice. Circle the letter of the most correct answer. Give an optional reason for each question for partial credit. If your answer for a question is correct, the reason will not be considered. If you think that none of the answers are correct, indicate your correct answer. Answer only 13 out of 15 questions. If you answer all 15 questions, the questions with the two lowest scores will be dropped. 3 points each.

1. (Python) Which of these assert statements will cause a failure in a unit test class?

   a. `self.assertEqual(9, 3 * 3)` b. `assertEqual(9, 2 ** 3)`
   c. `self.assertEqual(4, 2 * 2))` d. `assertEqual(6, 3 + 3)`

2. (Python) What is the output?

```
arr = [[3, 6, 5, 9], [8, 0, 1], [7, 4, 8, 2, 1]]
for r in arr:
    print(len(r), end=" ")
```

   a. `3 6 5 9 8 0 1 7 4 8 2 1` b. `3 8 7` c. `4 3 5` d. `23 9 22`

3. (Python) The method `get_factors` is supposed to return a list of all the factors of a positive integer. Which of these choices shows the method definition with statements in the correct order?

a.
```
def get_factors(number):
    factor_list = [1]
    for f in range(2, number):
        if number % f == 0:
            factor_list.append(f)
    factor_list.append(number)
    return factor_list
```

b.
```
def get_factors(number):
    for f in range(2, number):
        factor_list = [1]
        if number % f == 0:
            factor_list.append(f)
    factor_list.append(number)
    return factor_list
```

c.
```
def get_factors(number):
    factor_list = [1]
    for f in range(2, number):
        if number % f == 0:
            factor_list.append(f)
    return factor_list
    factor_list.append(number)
```

d.
```
def get_factors(number):
    factor_list = [1]
    if number % f == 0:
        for f in range(2, number):
            factor_list.append(f)
    factor_list.append(number)
    return factor_list
```

4. (Python) What is the output?

```
n = 1
s = "abcdefghijklmnopqrstuvwxyz"
while i <= 25:
    print(s[i], end="")
    n *= 2
```

a. `abdnp` b. `bceiq` c. `bep` d. `cdq`

5. (Python) What is the output?

```
print("cde" < "abc")
```

a. `-1` b. `1` c. `False` d. `True`

6. (Python) What is the constructor for the `PokerHand` class?
a. `__init__` b. `__repr__` c. `classify` d. `PokerHand`

7. (Python, Java) Which test is used to check if inheritance makes sense for two classes?
a. composition b. is-a c. has-a d. subordination

8. (Python) What is another name for the base class in an inheritance hierarchy?
a. derived class b. foundation class c. parent class d. subclass

9. What is the output?

```
lst = list( )
print(lst)
```

a. `""` b. `0` c. `[ ]` d. `{""}`

10. (Python) Which Python regular expression accepts a valid grade point average (GPA) rounded to two digits after the decimal point. A valid GPA must be between 0.00 and 4.00, inclusive.
a. `r"\A(0|1|2|3|4)\.\d{2}\Z"` b. `r"\A\d\.\d{2}\Z"`
d. `r"\A\d*\.\d{2}\Z"` d. `r"\A((0|1|2|3)\.\d{2})|(4\.00)\Z"`

11. (Python) What is the output from executing the file `test.py`:

```
#### Source code file a.py. ####
class A:
    def __init__(self, the_x):
        self.x = the_x
    def __str__(self):
        return f"%{str(self.x)}%"
    def augment(self):
        self.x *= 2
#### Source code file b.py. ####
from a import A
class B(A):
    def __init__(self, the_x, the_y):
        self.x = the_x
        self.y = the_y
    def __str__(self):
        return f"*{str(self.x + self.y)}*"
#### Source code file test.py. ####
from a import A
from b import B
a = A(6)
b = B(4, 8)
a.augment( )
b.augment( )
print(a, b)
```

A object a
`self.x`

B object b	
`self.x`	`self.y`

a. `%6% *4*` b. `%6% *4 8*` c. `%12% *16*` d. `%12% *8 16*`

12. (Python) What is the output?

```
import sqlite3
conn = sqlite3.connect("pets.db")
cur = conn.cursor( )
cur.execute("create table pets(" + \
    "name varchar(10), gender varchar(1), age integer);")
cur.execute("insert into pets values('Mugsie', 'M', 3);")
cur.execute("insert into kids values('Shadow', 'F', 4);")
cur.execute("insert into kids values('Bentley', 'M', 5);")
cur.execute("insert into kids values('Mimi', 'F', 2);")
conn.commit( )
cur.execute("select name, age from pets where gender = 'F';")
print(cur.fetchall( ))
conn.close( )
```

a. `[('Shadow', 4)]`
b. `[('Shadow', 4), ('Mimi', 2)]`
c. `[('Mugsie', 3), ('Bentley', 5)]`
d. `[('Mugsie', 3), ('Shadow', 4), ('Bentley', 5), ('Mimi', 2)]`

13. (Python) The lists `ids` and `names` are defined as

```
ids = [22222, 33333, 44444, 55555]
names = ["Alice", "Taylor", "Chloe", "Scott"]
```

The `make_dictionary` method returns a dictionary with keys in a list such as `ids` and values in a list such as `names`. Which of these choices has its statements in the correct order?

a.
```
def make_dictionary(keys, values):
    for i in range(0, len(keys)):
        dictionary[keys[i]] = values[i]
    return dictionary
    dictionary = { }
```

b.
```
def make_dictionary(keys, values):
    dictionary = { }
    for i in range(0, len(keys)):
        return dictionary
    dictionary[keys[i]] = values[i]
```

c.
```
def make_dictionary(keys, values):
    for i in range(0, len(keys)):
        dictionary = { }
        dictionary[keys[i]] = values[i]
    return dictionary
```

d.
```
def make_dictionary(keys, values):
    dictionary = { }
    for i in range(0, len(keys)):
        dictionary[keys[i]] = values[i]
    return dictionary
```

14. (Java) Which is a properly written constructor for a class `A.` The constructor initializes a string instance variable `s`.

```
a. public A(String theS)           b. public String A(String theS)
   {                                  {
       s = theS;                          s = theS;
   }                                  }
d. public void A(String theS)      d. public void A(String theS)
   {                                  {
       s = theS;                          theS = s;
   }                                  }
```

15. (Java) An object from which class is used to read from the keyboard or froma file?
    a. `IOInput` b. `IOReader` c. `Reader` d. `Scanner`

Part B: Find the errors in Python Source Code. There are about 10 total errors in the bridgehand.py and test.py files on this page. Correct the errors directly on pages 5 and 6. Do not recopy. (8 points penalty for recopying.) Correcting a pair of `( ), [ ], { }, ""`, or `''` only counts as one error. Correcting a pair of `( ), [ ], { },` double quotes, or single quotes only counts as one error. 10 points.
The source code files `card.py` and `deck.py` are shown on pages 9 and 10.

```
# Source code file bridgehand.py
from card import card
from deck imports Deck
class BridgeHand(Deck):

    def __init_(self, the_cards):
        self._cards = [ ]
        for card in the_cards:
            self._cards.append(card)

    # To value a bridge hand, add points for each
    # honor card, counting each Ace (rank 14) as 4 points,
    # each King (rank 13) as 3 points, each Queen (rank 12)
    # as 2 points, and each Jack (rank 11) as 1 point.
    # Use the dictionary d to do this.
    def point_count( ):
        total_point_count == 0
        d = [14: 4, 13: 3, 12: 2, 11: 1]
        for card in self._cards:
            if card in d:
                total_point_count += d[card.rank]
            return total_point_count

# Source code file test.py
import card
from bridgehand import BridgeHand
arr = [ Card(13, "S"), Card(11, "D"), Card(3, "H"),  Card(12, "H"),
        Card(5, "S"),  Card(13, "D"), Card(10, "S"), Card(12, "S"),
        Card(13, "S"), Card(11, "D"), Card(10, "H"), Card(8, "C")
        Card(14, "H") ]
bh = new BridgeHand(arr)
print(bh.point_count( ))
c = bh.deal
print(str(c))
```

Part C: Predict the Output. Predict the output of the test file `test1.py`, on Page 7. Justify your answer. 8 points.

```
# Source code file librarybook.py
from book import Book
class LibraryBook(Book):

    # super( ) is used to call methods from the base class.
    def __init__(self, the_author, the_title, the_catalog_number):
        super( ).__init__(the_author, the_title)
        self.catalog_number = the_catalog_number
        self._is_checked_out = False

    def __str__(self):
        return super( ).__str__( ) + \
            f"{self.catalog_number}\n{self._is_checked_out}\n"

    def check_out(self):
        self._is_checked_out = True

    def check_in(self):
        self._is_checked_out = False

    # Returns True or False, depending on whether
    # or not the book object is checked out
    def checked_out(self):
        return self._is_checked_out

# Source code file test1.py
from book import Book
from librarybook import LibraryBook

b1 = Book("War and Peace", "Tolstoy")
print(b1)

b2 = LibraryBook("Catcher in the Rye", "Salinger", 813.54)
print(b2)
b2.check_out( )
print(b2.checked_out( ))
b2.check_in( )
print(b2.checked_out( ))
```

Part D: Convert Traditional Test to Unit Test. Convert the traditional test in `test1.py` into a unit test. Convert `print` statements to statements like
`self.assertEqual(computed_value, expected value)`
Leave non-print statements as is. 8 points.

```
# Source code file test2.py
from book import Book
from librarybook import LibraryBook
import unittest

class MyUnitTestClass(unittest.TestCase):

    def test_1(self):
        b1 = Book("War and Peace", "Tolstoy")








    def test_2(self):
        b2 = LibraryBook("Catcher in the Rye", "Salinger", 813.54)










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

Part E: Find Errors in Java Source Code. There are about 10 total errors in the bridgehand.py and test.py files on this page. Correct the errors directly on pages 5 and 6. Do not recopy. (5 point penalty for recopying.) Correcting a pair of `( ), [ ], { }, ""`, or `''` only counts as one error. Correcting a pair of `( ), [ ], { },` double quotes, or single quotes only counts as one error. The source code files `card.py` and `deck.py` are shown on Pages 9 and 10. 10 points.

```
// Source code file Card.java. Correct errors in this class.
// A Card object is a standard poker or bridge playing card.
public Class Card
{
    public int rank;
    private String suit;

    public Card(int theRank, String theSuit)
    {
        rank = theRank;
        suit = theSuit;


    public int color( )
    {
        // Use equals method to compare String
        // objects instead of ==.  || means or in Java.
        if (suit.equals("C") || suit.equals("S")
        {
            return "black";
        }
        else
        {
            return "red";
        }
    }

    // Represent card as a string.
    // The symbols list entries "0" and "1" are dummy entries.
    public void String toString
    {
        String[ ] symbols = {"0", "1", "2", "3", "4", "5", "6", "7",
            "8", "9", "10", "J", "Q", "K", "A"};
        return symbols[rank] + suit;
    }
}
```

```
// This file tests the Card class on Page 8. Correct errors in this class.
public class TestCard
{
    public void static main(String args)
    {
        Card c1 = new Card(12, "S");
        System.out.println(c1.toString( ));
        System.out.println(c1.rank + " " + c1.suit + " " + c1.color( ));
        System.out.println( );

        Card c2 = new Card(14, 'H');
        System.out.println(c2);
        System.out.println(c2.rank + " " + c2.suit " " + c2.color( ));
    }
}
```

```
# Python file for Card class: card.py (no errors).
class Card:
    def __init__(self, the_rank, the_suit):
        self.rank = the_rank
        self.suit = the_suit
    def color(self):
        if self.suit == "C" or self.suit == "S":
            return "black"
        else:
            return "red"
    # Represent card as a string.
    # The symbols list entries "0" and "1" are dummy entries.
    def __str__(self):
        symbols = ["0", "1", "2", "3", "4", "5", "6", "7", \
            "8", "9", "10", "J", "Q", "K", "A"]
        return symbols[self.rank] + self.suit
```

```
# Python test script for Card class: testcard.py (no errors).
from card import Card
c1 = Card(12, "S")
print(c1)
print(c1.rank, c1.suit, c1.color( ))
print( )
c2 = Card(14, "H")
print(c2)
print(c2.rank, c2.suit, c2.color( ))
```

```
# Python file for Deck class (no errors).
# Implement a class that represents a deck
# of 52 playing cards, each card with a rank and suit.
from card import Card
from random import shuffle

class Deck:

    # Initialize deck to contain 52
    def __init__(self):
        self._cards = [ ]
        for suit in ["C", "D", "H", "S"]:
            for rank in range(2, 15):
                self._cards.append(Card(rank, suit))

    # Display deck as a str object.
    def __str__(self):
        output = ""
        for card in self._cards:
            output += " " + str(card)
        return output

    # Remove top card from the deck and return it.
    def deal(self):
        return self._cards.pop( )

    # Add a card to top of deck
    def add_to_top(self, the_card):
        self._cards.append(the_card)

    # Add a card to bottom of deck
    def add_to_bottom(self, the_card):
        self._cards.insert(0, the_card)

    # Return number of cards in deck
    def count(self):
        return len(self._cards)

    # Return True of deck is empty, False otherwise.
    def is_empty(self):
        return len(self._cards) == 0

    # Shuffle the cards
    def shuffle(self):
        shuffle(self._cards)
```