

BRONX COMMUNITY COLLEGE

of The City University of New York

DEPARTMENT OF MATHEMATICS and COMPUTER SCIENCE

Sample questions from CSI32 Final Exam

Directions: Answer all questions. You may refer to the textbook, handouts, notes taken in class, and use computer, but you may not discuss your work with other students. You cannot use cellphones.

Write answers and explanations clearly. The questions are worth the points indicated.

(24) Answer TRUE or FALSE, and **explain your answer**:

1. In Python language one can start appending (i.e. adding) elements to a list object even if it wasn't created/declared/instantiated beforehand.
2. A class in Python is *homogeneous* if its elements are of different types.
3. There is no way for a class to re-use all methods and attributes of some other class.
4. A class method has to have `self` as one of its parameters.
5. When defining a *constructor/initializer* for a class in Python, it should have same name as the class itself.

6. $3 \leq 10$ or $7 \leq 4$ and 'm' in 'Mother'

(find truth value of this statement)

comment: don't forget the order/precedence of logical operators

(25) Consider the following recursive definition of a Python's method, where n is integer

```
def conv(n):
    m = n//8
    if m != 0:
        return conv(m) + str(n%8)
    else:
        return str(n%8)
```

1. Describe what the method does. What is the result of the call `conv(92)`? Explain how the result was found.

2. Draw the trace of the call `conv(92)`.
(show function call and return values).

(16) Answer each of the questions

1. Find and show the *four* (syntax and other types of) errors in the following code. Explain how to fix them.

```
from cs1graphics import *

def main()
    paper = Canvas(800,600)

    r = Rectangle(30,60,100,200)
    r.setFillColor(blue)
    r.add(paper)

main()
```

2. Describe the output or the cause of the error. Explain.

```
['10', '11'] + '12,13,14,15'.split(',')
```

3. Use *list comprehension* to generate the list of values $[-1, 4, -9, 16, -25, 36, -49, 64, -81]$

(35) Let's recall the complex numbers:

A complex number is a number comprising a *real* and an *imaginary part*. It can be written in the form $a + bi$, where a and b are real numbers, and i is the standard *imaginary unit* with the property $i^2 = -1$. Below is a partial definition of a class, **Complex**. As you can see, constructor, addition, division and display methods are defined already, but multiplication, subtraction and negation operations are not.

```
class Complex:
    def __init__(self,r=0,im=0):
        self._r = r
        self._im = im

    def __str__(self):
        if self._r == 0 :
            return str(self._im) + 'i'
        elif self._im == 0:
            return str(self._r)
        else:
            if self._im < 0:
                return str(self._r) + str(self._im) + 'i'
            else:
                return str(self._r) + '+' + str(self._im) + 'i'

    def __add__(self, other):
        return Complex(self._r + other._r, self._im + other._im)

    def __mul__(self, other):
        pass

    def __sub__(self, other):
        pass

    def __truediv__(self, other):
        a,b = self._r, self._im
        c,d = other._r, other._im
        return Complex((a*c + b*d)/(c**2 + d**2), (b*c - a*d) / (c**2 + d**2))

    def __neg__(self):
        pass
```

In case you need rules for addition, subtraction, multiplication and division of complex numbers, here they are:

Let $r_1 = a + bi$, and $r_2 = c + di$, then

$$r_1 + r_2 = (a + bi) + (c + di) = (a + c) + (b + d)i$$

$$r_1 - r_2 = (a + bi) - (c + di) = (a - c) + (b - d)i$$

$$r_1 \cdot r_2 = (a + bi) \cdot (c + di) = (ac - bd) + (ad + bc)i$$

$$r_1 \div r_2 = \frac{a + bi}{c + di} = \frac{(a + bi) \cdot (c - di)}{(c + di) \cdot (c - di)} = \frac{(ac + bd) + (bc - ad)i}{c^2 + d^2}$$

1. Write the definition of the subtraction, multiplication and negation operations (i.e. `--sub--`, `--mul--`, `--neg--` methods)

2. Define *accessor methods* to access ('get') the real part of a complex number (i.e. `_r`), and the imaginary part of a complex number (i.e. `_im`): `getR`, `getIM`

3. Draw the Class Diagram, which contains all the above information about the `Complex` class, along with the accessor methods.