CSCI1200 Python Practice Exercises (Numerical Computation)

By: Dr. Ming Ming Tan.

These are self-practice exercises for beginners. These exercises are easier than the lab and exam questions. The purpose of these exercises is to ensure that you have mastered the minimum basics of programming.
You are asked to work out these exercises by hand, rather than implementing them.

Numerical Computation

1. What is the decimal integer represented by the binary string "1100"?
2. What is the decimal integer represented by the binary string "1101"?
3. What is the decimal integer represented by the binary string "1111"?
4. Write the binary string representation of integers from 0 to 8.
   0:
   1:
   2:
   3:
   4:
   5:
   6:
   7:
   8:
5. Consider the following code.

```
def test(text):
    d = 0
    n = len(text)
    for i in range(n):
      d+=2**(n-1-i)*int(text[i])
    return d
```

What is the output of:

(a) `test("001")`
(b) `test("010")`
(c) `test("011")`
(d) `test("100")`
(e) `test("101")`
(f) `test("110")`
(g) `test("111")`

6. Consider the following code.

```
def test(text):
    d = 0
    n = len(text)
    for i in range(n):
      d = 2*d + int(text[i])
    return d
```

What is the output of:

(a) `test("001")`
(b) `test("010")`
(c) `test("011")`
(d) `test("100")`
(e) `test("101")`
(f) `test("110")`
(g) `test("111")`

7. Complete the following function `test` which takes an input string `text` which represents an integer value in base 3, and returns the corresponding decimal value of `text`.
   Requirement: You are not allowed to import any libraries or use the built-in-functions to do the conversions.
   For example, `test("112")` should return
   $1 \times 3^2 + 1 \times 3^1 + 2 \times 3^0 = 9 + 3 + 2 = 14$.

```
def test(text):
```

8. Consider the following function.

```
def test(d):
  b = ""
  while d > 0:
    b = str(d % 2) + b
    d = d // 2
  return b
```

What is the output of :

(a) `test(1)`
(b) `test(2)`
(c) `test(3)`
(d) `test(4)`
(e) `test(5)`
(f) `test(6)`
(g) `test(7)`

9. Consider the following function.

```
def test(d):
  b = ""
  while d > 0:
    if d%2==0:
      b = "0" + b
    else:
      b = "1" + b
  return b
```

What is the output of :

(a) `test(1)`
(b) `test(2)`
(c) `test(3)`
(d) `test(4)`

(e) `test(5)`
(f) `test(6)`
(g) `test(7)`

10. Consider the following function.

```
def test(d):
  b = ""
  while d > 0:
    if d%2==0:
      b = b + "0"
    else:
      b = b + "1"
  return b[::-1]
```

What is the output of :

(a) `test(1)`
(b) `test(2)`
(c) `test(3)`
(d) `test(4)`
(e) `test(5)`
(f) `test(6)`
(g) `test(7)`

11a. Consider the following function.

```
def test(d):
  b = ""
  while d > 0:
    if d%2==0:
      b = b + "0"
    else:
      b = b + "1"
  return b
```

What is the output of :

(a) `test(1)`

(b) `test(2)`
(c) `test(3)`
(d) `test(4)`
(e) `test(5)`
(f) `test(6)`
(g) `test(7)`

11b. Which of the following describe the function `test` above:
(a) Given an integer `d`, `test(d)` returns the binary representation of `d` as a string.
(b) Given an integer `d`, `test(d)` returns the binary representation of `d` as a string in reversed order.

12. Complete the following function `test` which takes an input integer `d` and returns the base 3 representation of `d` as string.
    Requirement: You are not allowed to import any libraries or use the built-in-functions to do the conversions.
    For example, `test(14)` should return `"112"`.

```
def test(text):



```

13a. Consider the following code

```
0.1+0.1+0.1==0.3
```

The evaluated value is `False`. Write down the reason that might have resulted in the expression to be evaluate to `False` despite the fact that 0.1+0.1+0.1 should be equal to 0.3.

13b. Rounding errors might occur when we do arithmetic operations on floating point numbers. List down three strategies that we can use to avoid rounding errors.

14. We have learned that 0.1 can't be represented with finitely many number of binary digits. How about 0.2? If yes, what is the binary representation of 0.2. If no, why?

15. Consider the following code:

```
def verify_cube_root(r,n):
  return r**3==n
```

The output of `verify_cube_root(0.1, 0.001)` is `False`. Why?
