CSCI1200 Python Practice Exercises (Exhaustive Enumeration)

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.

Exhaustive Enumeration

1a. Consider the following code, where x is an integer and L is a list of integers.

```
def test(x, L):
  for element in L:
    if x==element:
        return True
    else:
        return False
```

What is the output of
(a) `test(3, [1,2,4])`
(b) `test(3, [1, 2, 3, 4])`
(c) `test(1, [1, 2, 3])`

1b. Consider the following code, where x is an integer and L is a list of integers.

```
def test(x, L):
  for element in L:
    if x==element:
        return True
    return False
```

What is the output of

(a) `test(3, [1,2,4])`

(b) `test(3, [1, 2, 3, 4])`

(c) `test(1, [1, 2, 3])`

1c. Consider the following code, where x is an integer and L is a list of integers.

```
def test(x, L):
  for element in L:
    if x==element:
        return True
  return False
```

What is the output of

(a) `test(3, [1,2,4])`

(b) `test(3, [1, 2, 3, 4])`

(c) `test(1, [1, 2, 3])`

2. Write a function `test` that takes an integer `x` and a list `L` as inputs and return `True` if `x` is in `L`, `False` otherwise. Requirement: You are not allowed to just return `x in L`, use iteration instead.

```
def test(x,L):
```

3. Write a function `test` that takes an integer `x` and a list `L` as inputs and return `True` if `x` is NOT in `L`, `False` otherwise. Requirement: You are not allowed to just return `x not in L`, use iteration instead.

```
def test(x,L):
```

4a. Consider the following code, where x is an integer and L is a list of integers.

```
def test(x, L):
  for element in L:
    if x%element==0:
      return True
    else:
      return False
```

What is the output of

(a) `test(14, [2,3,5])`

(b) `test(49, [2,3,5])`

(c) `test(14, [3,5, 7])`

4b. Consider the following code, where x is an integer and L is a list of integers.

```
def test(x, L):
  for element in L:
    if x%element==0:
      return True
    return False
```

What is the output of

(a) `test(14, [2,3,5])`

(b) `test(49, [2,3,5])`

(c) `test(14, [3,5, 7])`

4c. Consider the following code, where x is an integer and L is a list of integers.

```
def test(x, L):
  for element in L:
    if x%element==0:
      return True
  return False
```

What is the output of

(a) `test(14, [2,3,5])`

(b) `test(49, [2,3,5])`
(c) `test(14, [3,5, 7])`

5. Write a function `test` that takes an integer `x` and a list `L` and returns `True` if there is an element in `L` that can divide `x`.

```
def test(x,L):
```

6. Write a function `test` that takes an integer `x` and a list `L` and returns `True` if there is NO element in `L` that can divide `x`.

```
def test(x,L):
```

7. Consider the following code.

```
def test(x):
  for d in range(2,x):
      if x%d==0:
        return False
      return True
```

What is the output of
(a) `test(2)`
(b) `test(3)`

(c) `test(4)`
(d) `test(5)`

8. Consider the following code.

```
def test(x):
  for d in range(2,x):
      if x%d==0:
        return False
  return True
```

What is the output of
(a) `test(2)`
(b) `test(3)`
(c) `test(4)`
(d) `test(5)`

9. Write a function `test` that takes an integer `x` and returns `True` if `x` is a prime number.

```
def test(x)
```

10. Consider the following code.

```
def test(x):
  if x%2==0:
    return False
  for d in range(3,x,2):
      if x%d==0:
        return False
  return True
```

What is the output of

(a) `test(2)`

(b) `test(3)`

(c) `test(4)`

(d) `test(5)`

11. Consider the following code.

```
def test(x):
  if x%2==0:
    return False
  elif x%3==0:
    return False
  for d in range(5,x,2):
      if x%d==0:
        return False
  return True
```

What is the output of

(a) `test(2)`

(b) `test(3)`

(c) `test(42)`

(d) `test(15)`

(e) `test(17)`

12. Consider the following code.

```
def test(x):
  for r in range(1,x):
    if r**3==x:
      return r
  return -1
```

What is the output of

(a) `test(1)`

(b) `test(2)`

(c) `test(8)`

(d) `test(27)`
(e) `test(16)`

13. Consider the following code.

```
def test(x):
  for r in range(1,x+1):
    if r**3==x:
      return r
  return -1
```

What is the output of
(a) `test(1)`
(b) `test(2)`
(c) `test(8)`
(d) `test(27)`
(e) `test(16)`

14. Consider the following code.

```
def test(x):
  for r in range(1,x+1):
    if r**3==x:
      return r
    if r**3>x:
      break
  return -1
```

What is the output of
(a) `test(1)`
(b) `test(2)`
(c) `test(8)`
(d) `test(27)`
(e) `test(16)`

15. Consider the following code.

```
def test(x):
```

```
  for r in range(1,x+1):
    if r**3==x:
      return r
    if r**3<x:
      break
  return -1
```

What is the output of

(a) `test(1)`

(b) `test(2)`

(c) `test(8)`

(d) `test(27)`

(e) `test(16)`

16. Write the function `test(x)` that takes a positive integer `x` and returns the integer cube root of `x` if it exists, else returns -1.

```
def test(x):
```

16. Consider the following code.

```
def test(x):
  y = abs(x)
  for r in range(1,y+1):
    if r**3==y:
      return r
```

```
    if r**3>y:
      break
  return -1
```

What is the output of

(a) `test(1)`
(b) `test(2)`
(c) `test(8)`
(d) `test(27)`
(e) `test(16)`

17. Consider the following code.

```
def test(x):
  y = abs(x)
  for r in range(1,y+1):
    if r**3==y:
      if x<0:
        return -r
      else:
        return r
    if r**3>y:
      break
  return -1
```

What is the output of

(a) `test(1)`
(b) `test(2)`
(c) `test(8)`
(d) `test(27)`
(e) `test(16)`

18. Write the function `test(x)` that takes an integer `x` and returns the integer cube root of `x` if it exists, else returns -1. Notes the integer `x` can be negative,

```
def test(x):

```