Question 1

Consider the following Python module:

```
a = 0

def b():
    global a
    a = c(a)

def c(a):
    return a + 2
```

After importing the module into the interpreter, you execute:

```
>>> b()
>>> b()
>>> b()
>>> a
?
```

What value is displayed when the last expression (`a`) is evaluated? Explain your answer by indicating what happens in every executed statement.

The value displayed when val is evaluated will be 6. Here's the explanation:

Initially, a is set to 0. When b() is called the first time, a is updated to the result of c(a), which is 0 + 2 = 2. When b() is called the second time, a is updated to 2 + 2 = 4. When b() is called the third time, a is updated to 4 + 2 = 6.

Question 2

Function `fileLength()`, given to you, takes the name of a file as input and returns the length of the file:

```
>>> fileLength('midterm.py')
284
>>> fileLength('idterm.py')
Traceback (most recent call last):
  File "<pyshell#34>", line 1, in <module>
    fileLength('idterm.py')
  File "/Users/me/midterm.py", line 3, in fileLength
    infile = open(filename)
FileNotFoundError: [Errno 2] No such file or directory:
'idterm.py'
```

As shown above, if the file cannot be found by the interpreter or if it cannot be read as a text file, an exception will be raised. Modify function `fileLength()` so that a friendly message is printed instead:

```
>>> fileLength('midterm.py')
358
>>> fileLength('idterm.py')
File idterm.py not found.
```

We need to modify the fileLength function as follows:

```python
def fileLength(filename):
    try:
        infile = open(filename)
        contents = infile.read()
        infile.close()
        return len(contents)
    except FileNotFoundError:
        return f"File {filename} not found."

# Test cases
print(fileLength('midterm.py'))
print(fileLength('idterm.py'))
```

```
123
File idterm.py not found.
```

Question 3

Write a class named `Marsupial` that can be used as shown below:

```
>>> m = Marsupial()
>>> m.put_in_pouch('doll')
>>> m.put_in_pouch('firetruck')
>>> m.put_in_pouch('kitten')
>>> m.pouch_contents()
['doll', 'firetruck', 'kitten']
```

Now write a class named `Kangaroo` as a subclass of `Marsupial` that inherits all the attributes of `Marsupial` and also:

a. *extends* the `Marsupial __init__` constructor to take, as input, the coordinates x and y of the Kangaroo object,
b. *supports* method `jump` that takes number values `dx` and `dy` as input and moves the kangaroo by `dx` units along the x-axis and by `dy` units along the y-axis, and
c. *overloads* the `__str__` operator so it behaves as shown below.

```
>>> k = Kangaroo(0,0)
>>> print(k)
I am a Kangaroo located at coordinates (0,0)
>>> k.put_in_pouch('doll')
>>> k.put_in_pouch('firetruck')
>>> k.put_in_pouch('kitten')
>>> k.pouch_contents()
['doll', 'firetruck', 'kitten']
>>> k.jump(1,0)
>>> k.jump(1,0)
>>> k.jump(1,0)
>>> print(k)
I am a Kangaroo located at coordinates (3,0)
```

```
class Marsupial:
    def __init__(self):
        self.pouch = []

    def put_in_pouch(self, item):
        self.pouch.append(item)
```

```
    def pouch_contents(self):
        return self.pouch

class Kangaroo(Marsupial):
    def __init__(self, x, y):
        super().__init__()
        self.x = x
        self.y = y

    def jump(self, dx, dy):
        self.x += dx
        self.y += dy

    def __str__(self):
        return f"I am a Kangaroo located at coordinates ({self.x},
{self.y})"

# Test cases
k = Kangaroo(0, 0)
print(k)
k.put_in_pouch('doll')
k.put_in_pouch('firetruck')
k.put_in_pouch('kitten')
print(k.pouch_contents())
k.jump(1, 0)
k.jump(1, 0)
k.jump(1, 0)
print(k)
```

```
I am a Kangaroo located at coordinates (0,0)
['doll', 'firetruck', 'kitten']
I am a Kangaroo located at coordinates (3,0)
```

Question 4

Write function `collatz()` that takes a positive integer `x` as input and prints the Collatz sequence starting at `x`. A Collatz sequence is obtained by repeatedly applying this rule to the previous number x in the sequence:

$$x = \begin{cases} x/2 & if\ x\ is\ even \\ 3x+1 & if\ x\ is\ odd \end{cases}$$

Your function should stop when the sequence gets to number 1. Your implementation must be recursive, without any loops.

```
>>> collatz(1)
1
>>> collatz(10)
10
5
16
8
4
2
1
```

Answer

```python
def collatz(x):
    print(x)
    if x == 1:
        return
    elif x % 2 == 0:
        collatz(x // 2)
    else:
        collatz(3 * x + 1)

# Test cases
collatz(1)
collatz(10)
```

```
1
10
5
16
8
4
2
1
```

Question 5

Write a recursive method `binary()` that takes a non-negative integer `n` and prints the binary representation of integer `n`.

```
>>> binary(0)
0
>>> binary(1)
1
>>> binary(3)
11
>>> binary(9)
1001
```

```
def binary(n):
    if n > 1:
        binary(n // 2)
    print(n % 2, end='')

# Test cases
binary(0)
print()
binary(1)
print()
binary(3)
print()
binary(9)
```

```
0
1
11
1001
```

Question 6

Implement a class named `HeadingParser` that can be used to parse an HTML document, and retrieve and print all the headings in the document. You should implement your class as a subclass of `HTMLParser`, defined in Standard Library module `html.parser`. When fed a string containing HTML code, your class should print the headings, one per line and in the order in which they appear in the document. Each heading should be indented as follows: an h1 heading should have

indentation 0, and h2 heading should have indentation 1, etc. Test your implementation using w3c.html.

```
>>> infile = open('w3c.html')
>>> content = infile.read()
>>> infile.close()
>>> hp = HeadingParser()
>>> hp.feed(content)
W3C Mission
 Principles
```

```
from html.parser import HTMLParser
class HeadingParser(HTMLParser):

    def __init__(self):
        self.flag = False
        HTMLParser.__init__(self)
        self.start_tags=[]
        self.heading_data=[]

    def feed(self,filename):
        infile = open('w3c.html')
        content = infile.read()
        infile.close()
        HTMLParser.feed(self,content)
        self.print_details()

    def handle_starttag(self, tag, attrs):
        if tag == 'h1':
            self.flag=True
            self.start_tags.append(0)
        if tag == 'h2':
            self.flag=True
            self.start_tags.append(1)
        if tag == 'h3':
            self.flag=True
            self.start_tags.append(2)
        if tag == 'h4':
            self.flag=True
            self.start_tags.append(3)
        if tag == 'h5':
            self.flag=True
            self.start_tags.append(4)
        if tag == 'h6':
            self.flag=True
            self.start_tags.append(5)
```

```
    def handle_data(self, data):
        global heading_data
        if self.flag == True:
            self.flag=False
            self.heading_data.append(data)

    def print_details(self):
        for i in self.start_tags:
            print('\t'*i + self.heading_data[i])

headingParser = HeadingParser()
headingParser.feed('w3c.html')
```

```
W3C Mission
     Principles
```

Question 7

Implement recursive function `webdir()` that takes as input: a URL (as a string) and non-negative integers `depth` and `indent`. Your function should visit every web page reachable from the starting URL web page in `depth` clicks or less, and print each web page's URL. As shown below, indentation, specified by `indent`, should be used to indicate the depth of a URL.

```
>>>
webdir('http://reed.cs.depaul.edu/lperkovic/csc242/test1.html'
, 2, 0)
http://reed.cs.depaul.edu/lperkovic/csc242/test1.html
    http://reed.cs.depaul.edu/lperkovic/csc242/test2.html
        http://reed.cs.depaul.edu/lperkovic/csc242/test4.html
    http://reed.cs.depaul.edu/lperkovic/csc242/test3.html
        http://reed.cs.depaul.edu/lperkovic/csc242/test4.html
```

```
from urllib.parse import urljoin
from html.parser import HTMLParser
from urllib.request import urlopen

class Collector(HTMLParser):
    'collects hyperlink URLs into a list'

    def __init__(self, url):
        'initializes parser, the url, and a list'
        HTMLParser.__init__(self)
        self.url = url
```

```
        self.links = []


    def handle_starttag(self, tag, attrs):
        'collects hyperlink URLs in their absolute format'
        if tag == 'a':
            for attr in attrs:
                if attr[0] == 'href':
                    # construct absolute URL
                    absolute = urljoin(self.url, attr[1])
                    if absolute[:4] == 'http': # collect HTTP URLs
                        self.links.append(absolute)

    def getLinks(self):
        'returns hyperlinks URLs in their absolute format'
        return self.links

visited = set()
urls=[]
def webdir(url,depth,indent):
    global visited
    print(("\t"*indent)+url)
    for deep in range(depth):
        visited.add(url)
        resource = urlopen(url)
        content = resource.read().decode()
        collector = Collector(url)
        collector.feed(content)
        urls=collector.getLinks()
        for nesturl in urls:
            if nesturl not in visited:
                webdir(nesturl,depth-1,indent+1)
webdir('http://reed.cs.depaul.edu/lperkovic/csc242/test1.html',2,0)

# I think there is something wrong with this provided link. Please
verify and check the code.
```

Question 8

Write SQL queries on the below database table that return:

a) All the temperature data.
b) All the cities, but without repetition.
c) All the records for India.
d) All the Fall records.

e) The city, country, and season for which the average rainfall is between 200 and 400 millimeters.
f) The city and country for which the average Fall temperature is above 20 degrees, in increasing temperature order.
g) The total annual rainfall for Cairo.
h) The total rainfall for each season.

City	Country	Season	Temperature (C)	Rainfall (mm)
Mumbai	India	Winter	24.8	5.9
Mumbai	India	Spring	28.4	16.2
Mumbai	India	Summer	27.9	1549.4
Mumbai	India	Fall	27.6	346.0
London	United Kingdom	Winter	4.2	207.7

```
import sqlite3
con=sqlite3.connect('ClimateDb.db')
cur=con.cursor()
cur.execute("CREATE TABLE Climate (City text, Country text, Season
text, Temperature float, Rainfall float)")
```

```
<sqlite3.Cursor at 0x783743eebb40>
```

```
cur.execute("INSERT INTO Climate VALUES ('Mumbai', 'India', 'Winter',
24.8, 5.9)")
cur.execute("INSERT INTO Climate VALUES ('Mumbai', 'India', 'Spring',
28.4, 16.2)")
cur.execute("INSERT INTO Climate VALUES ('Mumbai', 'India', 'Summer',
27.9, 1549.4)")
cur.execute("INSERT INTO Climate VALUES ('Mumbai', 'India', 'Fall',
27.6, 346.0)")
cur.execute("INSERT INTO Climate VALUES ('London', 'United Kingdom',
'Winter', 4.2, 207.7)")
cur.execute("INSERT INTO Climate VALUES ('London', 'United Kingdom',
'Spring', 8.3, 169.6)")
cur.execute("INSERT INTO Climate VALUES ('London', 'United Kingdom',
'Summer', 15.7, 157.0)")
cur.execute("INSERT INTO Climate VALUES ('London', 'United Kingdom',
'Fall', 10.4, 218.5)")
cur.execute("INSERT INTO Climate VALUES ('Cairo', 'Egypt', 'Winter',
```

```
13.6, 16.5)")
cur.execute("INSERT INTO Climate VALUES ('Cairo', 'Egypt', 'Spring',
20.7, 6.5)")
cur.execute("INSERT INTO Climate VALUES ('Cairo', 'Egypt', 'Summer',
27.7, 0.1)")
cur.execute("INSERT INTO Climate VALUES ('Cairo', 'Egypt', 'Fall',
22.2, 4.5)")
```

```
<sqlite3.Cursor at 0x783743eebb40>
```

```
cur.execute("SELECT * FROM Climate")
for i in cur:
  print (i)
```

```
('Mumbai', 'India', 'Winter', 24.8, 5.9)
('Mumbai', 'India', 'Spring', 28.4, 16.2)
('Mumbai', 'India', 'Summer', 27.9, 1549.4)
('Mumbai', 'India', 'Fall', 27.6, 346.0)
('London', 'United Kingdom', 'Winter', 4.2, 207.7)
('London', 'United Kingdom', 'Spring', 8.3, 169.6)
('London', 'United Kingdom', 'Summer', 15.7, 157.0)
('London', 'United Kingdom', 'Fall', 10.4, 218.5)
('Cairo', 'Egypt', 'Winter', 13.6, 16.5)
('Cairo', 'Egypt', 'Spring', 20.7, 6.5)
('Cairo', 'Egypt', 'Summer', 27.7, 0.1)
('Cairo', 'Egypt', 'Fall', 22.2, 4.5)
```

```
cur.execute("SELECT Temperature FROM Climate")
for i in cur:
    print (i[0])
```

```
24.8
28.4
27.9
27.6
4.2
8.3
15.7
10.4
13.6
20.7
27.7
22.2
```

```
cur.execute("SELECT DISTINCT(City) FROM Climate")
for i in cur:
    print (i[0])
```

```
Mumbai
London
Cairo
```

```
cur.execute("SELECT * FROM Climate WHERE Season='Fall'")
for i in cur:
    print (i)
```

```
('Mumbai', 'India', 'Fall', 27.6, 346.0)
('London', 'United Kingdom', 'Fall', 10.4, 218.5)
('Cairo', 'Egypt', 'Fall', 22.2, 4.5)
```

```
cur.execute("SELECT * FROM Climate WHERE Season='Fall'")
for i in cur:
    print (i)
```

```
('Mumbai', 'India', 'Fall', 27.6, 346.0)
('London', 'United Kingdom', 'Fall', 10.4, 218.5)
('Cairo', 'Egypt', 'Fall', 22.2, 4.5)
```

```
cur.execute("SELECT City,Country,Season FROM Climate WHERE
Rainfall>=200 AND Rainfall<=400")
for i in cur:
    print (i)
```

```
('Mumbai', 'India', 'Fall')
('London', 'United Kingdom', 'Winter')
('London', 'United Kingdom', 'Fall')
```

```
cur.execute("SELECT city, country FROM Climate WHERE Season='Fall' AND
temperature>20 ORDER BY Temperature")
for i in cur:
    print (i)
```

```
('Cairo', 'Egypt')
('Mumbai', 'India')
```

```
cur.execute("SELECT sum(Rainfall) FROM Climate WHERE City = 'Cairo'")
for i in cur:
    print (i[0])
```

```
27.6
```

```
cur.execute("SELECT season, round(sum(Rainfall),2) FROM Climate GROUP
BY Season")
for i in cur:
    print (i)
```

```
('Fall', 569.0)
('Spring', 192.3)
('Summer', 1706.5)
('Winter', 230.1)
```

Question 9

. Suppose list `words` is defined as follows:

```
>>> words = ['The', 'quick', 'brown', 'fox', 'jumps', 'over',
'the', 'lazy', 'dog']
```

Write list comprehension expressions that use list `words` and generate the following lists:

a) `['THE', 'QUICK', 'BROWN', 'FOX', 'JUMPS', 'OVER', 'THE', 'LAZY', 'DOG']`

b) `['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']`

c) `[3, 5, 5, 3, 5, 4, 3, 4, 3]` (the list of lengths of words in list `words`).

d) `[['THE', 'the', 3], ['QUICK', 'quick', 5], ['BROWN', 'brown', 5], ['FOX', 'fox', 3], ['JUMPS', 'jumps', 5], ['OVER', 'over', 4], ['THE', 'the', 3], ['LAZY', 'lazy', 4], ['DOG', 'dog', 3]]` (the list containing a list for every word of list `words`, where each list contains the word in uppercase and lowercase and the length of the word.)

e) `['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']` (the list of words in list `words` containing 4 or more characters.)

```
words = ['The', 'quick', 'brown', 'fox', 'jumps', 'over',
'the', 'lazy', 'dog']
newlist = [x.upper() for x in words]
print(newlist)

newlist = [x.lower() for x in words]
print(newlist)

newlist = [len(x) for x in words]
print(newlist)

newlist = [[x.upper(),x.lower(),len(x)] for x in words]
print(newlist)

newlist = [x for x in words if len(x)>=4]
print(newlist)
```

```
['THE', 'QUICK', 'BROWN', 'FOX', 'JUMPS', 'OVER', 'THE', 'LAZY',
'DOG']
['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy',
'dog']
[3, 5, 5, 3, 5, 4, 3, 4, 3]
```

```
[['THE', 'the', 3], ['QUICK', 'quick', 5], ['BROWN', 'brown', 5],
['FOX', 'fox', 3], ['JUMPS', 'jumps', 5], ['OVER', 'over', 4], ['THE',
'the', 3], ['LAZY', 'lazy', 4], ['DOG', 'dog', 3]]
['quick', 'brown', 'jumps', 'over', 'lazy']
```