Problem Set 2

# Question 1

## Output

6

## Code

1. a = 0
2. def b():
3. global a
4. a = c(a)
5. def c(a):
6. return a + 2

Execution steps:

1. b():
   1. First we call the function b(), inside it accesses the variable ‘a’ globally (not in local scope). So, any changes done to it will reflect outside the scope of function.
   2. Then we call the function c(a), which takes a variable ‘a’ as the parameter and returns the value ‘a’+2.
   3. Value returned from function c(a) is stored in the global variable ‘a’.

So, in each calling of the function b() we increment the value of variable ‘a’ by 2.

Since we have called the function b() 3 times and initial value of ‘a’ was 0, the final value of ‘a’ becomes 6.

1. a: Then we print the value of variable ‘a’

Output: 6

# Question 2

Correct code is given in the folder with the name **Filelength.py**

## Code

1. def file\_length(file\_name):
2. try:
3. file = open(file\_name)
4. contents = file.read()
5. file.close()
6. print(len(contents))
7. except:
8. print("File ", file\_name, "not found.")

To add this functionality, we used the ‘try’ ‘except’ block to handle the exception raised. So, whenever we fail to open a file, the execution of code within the ‘try’ block are stopped and the control jumps to the ‘except’ block.

# Question 3

## Code

class Marsupial:

def \_\_init\_\_(self):

self.pouch\_list = list()

def put\_in\_pouch(self, pouch\_item):

self.pouch\_list.append(pouch\_item)

def pouch\_contents(self):

return self.pouch\_list

class Kangaroo(Marsupial):

def \_\_init\_\_(self , x\_coord, y\_coord):

super().\_\_init\_\_()

self.x\_coord = x\_coord

self.y\_coord = y\_coord

def jump(self, dx, dy):

self.x\_coord += dx

self.y\_coord += dy

def \_\_str\_\_(self):

return "I am a Kangaroo located at coordinates (" + str(self.x\_coord) + ","+ str(self.y\_coord) + ")"

(Code is within **Question3.py)**

* The Kangaroo class is the subclass of Marsupial class, so Kangaroo class can inherit the methods and attributes present in Marsupial.
* We have overridden the \_\_str\_\_ method in Kangaroo class to provide a custom representation for any Kangaroo objects.
* We use super() function to call the constructor of the superclass Marsupial from the Kangaroo class.

# Question 4

## Code

def collatz(x: int):

# Base condition

if(x == 1):

print(x)

return

# Sub problems

print(x)

if x%2 == 0:

# x is even

collatz(x//2)

else:

# x is odd

collatz(3\*x + 1)

(Code is within **Question4.py)**

**Base case:** When x == 1 then we simply print 1 and return from the collatz() function.

**Subproblems:**

* When x is even: we update the new x as x/2 (Note: since x is even so x//2 and x/2 are equivalent), and then call the subproblem with the updated x.
* When x is odd: we update the new x as 3x+1, and then call the subproblem with the updated x

# Question 5

## Output

## Code

def binary(num: int):

if num < 0:

print("Provide a non-negative number")

return

if num == 0:

return "0"

binary\_out = ""

last\_bit = num % 2

# get remaining representation

sub\_binary\_out = ""

if num // 2 != 0:

# to avoid printing 0 twice, since the base case handles num == 0

sub\_binary\_out = binary(num//2)

binary\_out = sub\_binary\_out + str(last\_bit)

return binary\_out

(Code is within **Question5.py)**

First check of num < 0 is done in order to prevent someone from entering negative numbers.

Base case: if num == 0 then we simply return 0

We then check the last bit of the number in binary by performing num%2.

If the number is greater than 1, we call the subproblem by ‘binary(num//2)’ and it returns the binary representation of num//2.

After this we simply append the last bit of current number on the result from ‘binary(num//2)’.

The resulting string is binary representation of the number.

# Question 8

Let’s assume the table name is: **WeatherReport**

1. SELECT Temperature

FROM WeatherReport;

1. SELECT DISTINCT City

FROM WeatherReport;

1. SELECT \*

FROM WeatherReport

WHERE Country =” India”;

1. SELECT \*

FROM WeatherReport

WHERE Season =” Fall”;

1. SELECT City, Country, Season

FROM WeatherReport

GROUP BY City, Country, Season

HAVING AVG(Rainfall) BETWEEN 200 and 400;

1. SELECT City, Country

FROM WeatherReport

WHERE Season = “Fall”

GROUP BY City, Country

HAVING AVG(temperature) > 20;

1. SELECT SUM(Rainfall) as Total\_Annual\_Rainfall\_Cairo

FROM WeatherReport

WHERE City = ”Cairo”;

1. SELECT Season, SUM(Rainfall) as Total\_Rainfall

FROM WeatherReport

Group by Season;

# Question 9

## Output

## Code

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

**# Part (a)**

part\_a\_words = [word.upper() for word in words]

print("Part (a) output: " , part\_a\_words)

**# Part (b)**

part\_b\_words = [word.lower() for word in words]

print("Part (b) output: " , part\_b\_words)

**# Part (c)**

part\_c\_words = [len(word) for word in words]

# print("Part (c) output: " , part\_c\_words)

**# Part (d)**

part\_d\_words = [[word.upper(), word.lower(), len(word)] for word in words]

print("Part (d) output: " , part\_d\_words)

**# Part (e)**

part\_e\_words = [word for word in words if len(word) >= 4]

print("Part (e) output: " , part\_e\_words)

(Code is within **QUESTION9.Ipynb)**