Q. 1: Java and Python

A. Provide at least 4 points comparison between Java and Python.

Java	Python
Statically typed	Dynamically typed
Verbose	Terse (aka concise)
Not compact	Compact
Own file for each top level class	Multiple classes in a single file
Uses checked exceptions	Need not catch/throw exceptions in every method
Weak support for string handling	Good support for string handling
Heavy use of parenthesis	Minimal use of parenthesis

B. A binary gap within a positive integer N is any maximal sequence of consecutive zeros that is surrounded by ones at both ends in the binary representation of N. For example, number 9 has binary representation 1001 and contains a binary gap of length 2. The number 529 has binary representation 1000010001 and contains two binary gaps: one of length 4 and one of length 3. The number 20 has binary representation 10100 and contains one binary gap of length 1. The number 15 has binary representation 1111 and has no binary gaps. The number 32 has binary representation 100000 and has no binary gaps.

Write a function binaryGap (N) that, given a positive integer N, returns the length of its longest binary gap. The function should return 0 if N doesn't contain a binary gap. For example, given N = 1041 the function should return 5, because N has binary representation 10000010001 and so its longest binary gap is of length 5. Given N = 32 the function should return 0, because N has binary representation '100000' and thus no binary gaps.

You may write your solution in Java or Python depending on your choice.

```
def solution(N):
    cnt = 0
    result = 0
    found_one = False

    i = N

    while i:
```

```
        if i & 1 == 1:
            if (found_one == False):
                found_one = True
            else:
                result = max(result,cnt)
            cnt = 0
        else:
            cnt += 1
        i >>= 1

    return result
```

Q. 2. Python Object Oriented Programming: Implement a Python class clock. The class Clock simulates the tick-tack of a clock. An instance of this class contains the time, which is stored in the attributes self.hours, self.minutes and self.seconds. Complete the following methods of class clock:

```
def __init__(self, hours, minutes, seconds):
    """
    The parameters hours, minutes and seconds have to be integers and must satisfy the
    following equations: 0 <= h < 24, 0 <= m < 60, 0 <= s < 60. An exception is thrown
    if the values are outside range.
    """
      if type(hours) == int and 0 <= hours and hours < 24:
          self._hours = hours
      else:
          raise TypeError("Hours have to be integers between 0 and 23!")
      if type(minutes) == int and 0 <= minutes and minutes < 60:
          self.__minutes = minutes
      else:
          raise TypeError("Minutes have to be integers between 0 and 59!")
      if type(seconds) == int and 0 <= seconds and seconds < 60:
          self.__seconds = seconds
      else:
          raise TypeError("Seconds have to be integers between 0 and 59!")
def tick(self):
    """
    This method lets the clock "tick", this means that the internal time will be
    advanced by one second.
```

```
        """
                if self.__seconds == 59:
            self.__seconds = 0
            if self.__minutes == 59:
                self.__minutes = 0
                if self._hours == 23:
                    self._hours = 0
                else:
                    self._hours += 1
            else:
                self.__minutes += 1
          else:
            self.__seconds += 1
def __str__(self):
        """
        Prints the time in the format HH:MM:SS.
        """
return "{0:02d}:{1:02d}:{2:02d}".format(self._hours,
                                                self.__minutes,
                                                self.__seconds)

          Examples:
          >>> x = Clock(12,59,59)
          >>> print(x)
          12:59:59
          >>> x.tick()
          >>> print(x)
          13:00:00
          >>> x.tick()
          >>> print(x)
          13:00:01
          """
```

Q. 3. Python programming

A. Write a Python program for binary search for an ordered list. The program will return index of the number if it exists in the list otherwise will return -1 if the number does not exist.

```
Ordered_binary_Search([0, 1, 3, 8, 14, 18, 19, 34, 52], 3) -> 2
Ordered_binary_Search([0, 1, 3, 8, 14, 18, 19, 34, 52], 17) -> -1
```

```
def binarySearch(lst, key):

      low = 0

      high = len(lst) - 1
      while high >= low:
      mid = (low + high) // 2
      if key < lst[mid]:
            high = mid - 1
      elif key == lst[mid]:
            return mid
      else:
            low = mid + 1
      # Now high < low, key not found
      return - 1
```

B. Write a Python function to create a dictionary from a string. Note: Track the count of the letters from the string.
Sample string: 'w3resource'
Expected output: {'3': 1, 's': 1, 'r': 2, 'u': 1, 'w': 1, 'c': 1, 'e': 2, 'o': 1}

```
str1 = 'w3resource'

my_dict = {}

for letter in str1:

      my_dict[letter] = my_dict.get(letter, 0) + 1

print(my_dict)
```

Q. 4. Python program output

A. What is the output of the following Python code?

```
dictionary = {'GFG': 'geeksforgeeks.org',
              'google': 'google.com',
```

```
                'facebook': 'facebook.com'
                }
del dictionary['google'];
for key, values in dictionary.items():
    print(key)
dictionary.clear();
for key, values in dictionary.items():
    print(key)
```

Output:
GFG
Facebook

B. What is the output of the following Python code?

```
T = (1, 2, 3, 4, 5, 6, 7, 8)
print(T[T.index(5)], end = " ")
print(T[T[T[6]-3]-6])
```

Output:

5 8

C. What is the output of the following Python code?

```
from math import sqrt
L1 = [x**2 for x in range(10)][9]
L1 += 19
print(sqrt(L1), end = " ")
L1 = [x**2 for x in reversed(range(10))][0]
L1 += 16
print(int(sqrt(L1)))
```

Output:
10.0 4

D. What is the output of the following Python code?

```
L1 = [1, 1.33, 'GFG', 0, 'NO', None, 'G', True]
val1, val2 = 0, ''
for x in L1:
    if(type(x) == int or type(x) == float):
        val1 += x
    elif(type(x) == str):
        val2 += x
    else:
        break
print(val1, val2)
```

Output:
2.33 GFGNO

E. What is the output of the following Python code?

```
class A(object):
    val = 1

class B(A):
    pass

class C(A):
    pass

print (A.val, B.val, C.val)
B.val = 2
print (A.val, B.val, C.val)
A.val = 3
print (A.val, B.val, C.val)
```

Output:

1 1 1

1 2 1

3 2 3

4 Consider the following class Guitar. Write output of each of the print statements following the class.

```
class Guitar:
    # Construct a guitar object
    def __init__(self, id, numStrings = 12, price = 1000):
        self.id = id
        self.numStrings = 12
        self.price = price

    def getId(self):
        return self.id
    def getNumStrings(self):
        return self.numStrings
    def updatePrice(self, price):
        self.price = price
    def getPrice(self):
       return self.price
    def __str__(self):
        return "Guitar: numStrings = " + str(self.numStrings) + " price = " +
str(self.price) + " id = " + str(self.id)

guitarA = Guitar(123)
print(guitarA)

guitarB = Guitar(234, 8)
```

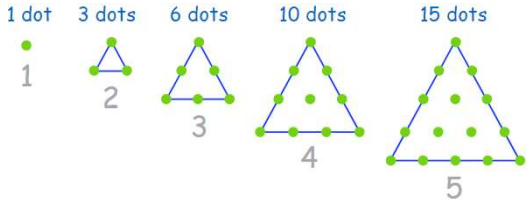
```
print(guitarB)

guitarC = Guitar(345, 8, 1500)
print(guitarC)

guitarD = Guitar(456, price = 2000)
print(guitarD)

guitarA.updatePrice(2500)
print(guitarA)
```

Q. 5: The **triangle numbers series** is generated by creating triangles of progressively larger size:



Write a function with the following definition to return a list of n elements in a triangle number series where n is the number of terms in a series. E.g. triangleSeries(6) should return a list [1, 3, 6, 10, 15, 21].

Q. 6. Write a password generator in Python where the password is 8 characters long. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password. Hint: gYou can randomly select an uppercase character using a function random.choice ("ABCDEFGHIJKLMNOPQRSTUVWXYZ")

Q. 7. Write a function, *acronym*, that creates an acronym from the first letter of each long word in a list, where a long word is any word with more than three letters. The acronym function takes an optional argument that tells it to include the first letter of all words, but short words (three or less letters) should not be capitalized.

```
>>> acronym('department of motor vehicles')
'DMV'
>>> acronym('department of justice', True)
'DoJ'
```

```
# version 2 of function
def acronym(phrase, include_shorts=False):
    result = ''
```

```
        words = phrase.split()
        for w in words:
        if len(w) > 3:
              result += w.upper()[0]
        elif include_shorts:
              result += w.lower()[0]
        return result
print(acronym('State University of New York at Stony Brook'))
print(acronym('State University of New York at Stony Brook', True))
```

Q. 8. (Classes and OOP)

Create a class called `Worker`. *Worker* holds information on a factory worker in a company. The information includes the worker's full name, hourly rate, hours in a standard week and hours in an extended week.

A worker earns their normal hourly rate for the number of hours in a standard week. If they work more hours, for the extra hours, they earn 1.5 times their hourly rate. Finally, if they work beyond the number of hours in the extended week, any hours over that number are paid at 2 times the hourly rate.

The class must have an __init__ method to build the object given the worker's name, hourly rate, standard hours, and extended hours.

You must also write a `calculatePay()` method that takes the number of hours worked that week and returns the amount of pay in US dollars.

Example: If Joe Cool has a standard work week of 40 hours, an extended week of 50 hours, and wage of 18.50 per hour, his pay for 55 hours would be:

40 * 18.50 + 10 * 18.50 * 1.5 + 5 * 18.50 * 2 = 1202.50

So creating a *Worker* object to compute this would look like:

```
w = Worker ("Joe Cool", 40, 50, 18.50)
print (w.name + " earned $" + str(w.calculatePay(55)) + " for 55 hours.")
```

Write the class along with the constructor and the calculatePay() method.

```
class Worker:

        def __init__(self, name, regular, extended, rate):
            self.name = name;
            self.regular = regular;
            self.extended = extended;
            self.rate = rate;

        def calculatePay(self, hours):
            if hours > self.extended:
                high = hours - self.extended
```

```
    over = self.extended - self.regular
    base = self.regular
elif hours > self.regular:
    over = hours - self.regular
    base = regular
else:
    base = hours
    over = 0
    high = 0
pay = (self.regular * self.rate) + (over * self.rate * 1.5) + (high * self.rate * 2);
return pay
```

1. Each of the following Python code examples contains a bug. Write the line number containing the bug and then show how to fix it. You may only change **one** line to make the fix.

   a. (4 pts) The function calculate(numbers) takes a list of integers and returns the sum of all the integers that are less than 10.

```
1. def calculate(numbers):
2.     total = 0
3.     for i in numbers:
4.         if i < 10:
5.             total += i
6.         return total
```

   Line _____6 contains a bug.

   It should be _____indented like the for loop .

   b. (4 pts) The function highest_divisor(number) returns the highest divisor for a number that is smaller than the number itself. For example, the highest divisor of 20 would be 10.

```
1. def highest_divisor(number):
2.     for i in range(number-1,0,-1):
3.         if number % i == 0:
4.             return number
5.     return None
```

   Line _____5 contains a bug.

It should be __return i_____.

c. (4 pts) the function count_fours(number) returns the amount of the digit '4' that are present in a positive integer (and returns 0 if the number is negative). For example 48643 would return 2 since it has two 4's, whereas 378 would return 0.

```
1. def count_fours(number):
2.     count = 0
3.     while num > 0:
4.         if num % 4 == 0:
5.             count += 1
6.         num //= 10
7.     return count
```

Line _____4 contains a bug.

It should be _____if num % 10 == 4:_____.

Complete the below_value(nums, max_value) function that returns a list of all values in list nums that are below the max_value. For example, if nums is [31, 5, 71, 53, 40, 17] and max_value is 40, then the function returns [31, 5, 17].

```
def below_value(nums, max_value):

    below_nums = []
    for n in nums:
       if n < max_value:
          below_nums.append(n)
    return below_nums
```

Write a Python function that takes a sentence as an input and returns the average word length in the sentence. A word may also include punctuation characters. The Python split() method which returns a list of strings after breaking the given string on whitespace characters may be useful here.

```
>>>average_word_length("Pursue what catches your heart, not what catches
your eyes.")
5.0
>>>average_word_length("Do not fear failure but rather fear not trying.")
4.333
```

```
def average_word_length(sentence):

    words = sentence.split()
    totalLetters = 0
    for word in words:
        totalLetters += len(word)
    return totalLetters / len(words)
```

Complete the get_fee() function below, which takes two (non-negative, numerical) arguments: the current account balance and the total number of electronic payments made this month. The function returns a new number representing this month's fee, which is calculated as follows:

d. The bank charges $5 automatically for every account every month.
e. If the account balance is less than $500.00 (after applying the $5 fee from the previous step), the bank adds an additional $20 charge to the fee.
f. Finally, the bank adds an additional charge based on the number of electronic payments made. This charge varies based on the total number of payments, according to the list below. The same fee is charged for all of the payments made that month. For example, if no payments were made this month, then this electronic payment charge will be 0, and if 25 payments were made, then the electronic payment charge will be $2.00:
    - 1–19 electronic payments: $0.10 each
    - 20–39 electronic payments: $0.08 each
    - 40 or electronic payments: $0.05 each

```
def get_fee(balance, num_payments):


        fee = 5
        if (balance - fee) < 500:
            fee += 20

        if num_payments < 20:
            fee += num_payments * 0.1
        elif num_payments < 40:
            fee += num_payments * 0.08
        else:
            fee += num_payments * 0.05
        return fee
```

1. Consider the following Python code fragment:

```
product = 0

for i in range(1, x):
    product = product * i

print(product)
```

If the variable x is initialized to 5, this code will:

A. print the value 5
B. print the value 24
C. print the value 120
D. print the value 0
E. print an error message

2. Which of the following is **NOT** required for an algorithm?

A. A precise statement of the starting conditions
B. A specification of the algorithm's termination condition
C. An electronic computing platform
D. A series of simple yet detailed steps
E. All of the above are required to create and implement an algorithm

3. The value of the arithmetic expression 5 - 3 * 2 is

A. -1
B. 4
C. 16
D. 0
E. None of the above

4. The value of the arithmetic expression 17 // 5 is A. 3.4

B. 3
C. 4
D. 5
E. None of the above

5. The value of the arithmetic expression 14 / 3 is

   A. 4

   B. 2

   C. 4.6667

   D. 0.667

   E. None of the above

6. Which of the following is **NOT** a valid Python arithmetic operator?

   A. *

   B. **

   C. /

   D. //

   E. **#**

7. Which of the following is **NOT** a legal variable name in Python?

   A. length

   B. CaptainAmerica

   C. _measure

   D. 99balloons

   E. None of the above; all four are valid variable names

8. What output will the following Python code fragment produce?

```
def mystery (f):
    f = f * 2
    print(f)

f = 5
mystery(f)
print(f)
```

   A. 5
      5

   B. 5
      10

   C. 10
      10

   D. 10
      5

   E. Nothing; an error message will be printed instead

9. Suppose that the variable income has the value 4001. What will be displayed by the following code?

```
if income  > 3000:
      print("Income is greater than  3000")
if income > 4000:
      print ("Income is greater than 4000")
```

A. Nothing will be displayed

B. *Income is greater than 3000*

C. *Income is greater than 3000* followed by *Income is greater than 4000*

D. *Income is greater than 4000*

E. *Income is greater than 4000* followed by *Income is greater than 3000*

10. Suppose that the variable income has the value 4500. What will be displayed by the following code?

```
if income > 3000:
      print("Income is greater than 3000")
elif income > 4000:
      print ("Income is greater than 4000")
```

A. Nothing will be displayed

B. *Income is greater than 3000*

C. *Income is greater than 3000* followed by *Income is greater than 4000*

D. *Income is greater than 4000*

E. *Income is greater than 4000* followed by *Income is greater than 3000*

11. Which of the following is the correct "does not equal" operator in Python?

A. !==

B. !=

C. =/=

D. !

E. None of the above

12. Given the Python statement below, what value would be returned by len(s)?

```
s = "I'm a little teapot!"
```

A. 15

B. 17

C. 18

D. 20

E. 22

13. The index of the last character in the Python string text is equal to:

A. 0
B. len(text) - 1
C. len(text)
D. text - 1
E. None of the above

14. Consider the following Python code fragment. What values will it print?

```
for x in range(1, 11, 3):
    print(x)
```

A. 1, 3, 6, 9
B. 1, 4, 7, 10
C. 2, 5, 8
D. 2, 5, 8, 11
E. 3, 6, 9

15. Assume that the string "candy" has been assigned to the Python variable c. What will the following Python code fragment print?

```
for i in range(4, 1, -1):
    print(c[i], end="")
print()
```

A. cand
B. andy
C. ydn
D. ydna
E. ydnac

16. What value will the following Python function return for the input "pandemonium"?

```
def mystery(text):
    result = ""
    for x in text:
        if x == 'p':
            result += "pop"
        elif x == 'm':
            result = ""
        else:
            result += x
    return result
```

A. popandemonium
B. pande
C. pandemonium
D. "" (the empty string)
E. pandeoniu

17. The for loop below is designed to add up all of the even integers (and *only* the even integers) in the range 6 through 104 (inclusive) and save the result into a variable named total. What code fragment should go in the blank space below?

```
total = 0

_____ : # your code goes here
    total = total + x
```

A. for x in range(6, 105)
B. for x in range(6, 104, 2)
C. for x in range(6, 105, 2)
D. None of the above

18. Given two Python strings defined as follows:

```
s1 = "heuristic"
s2 = "algorithm"
```

Which of the following expressions will return the value "urgor"?

A. s1[2:3] + s2[2:5]
B. s1[2:4] + s2[2:4]
C. s1[3:5] + s2[3:6]
D. s1[2:4] + s2[2:5]
E. None of the above

19. Consider the following function:

```
def blah(n):
    d = 0
    while n > 0:
        d = d * 10
        d = d + n % 10n
        = n // 10
    return d
```

What value will the function call blah(30792) return?

A. 27903

B. 32097

C. 29730

D. 29703

E. None of the above

20. What is the result of translating the base 10 value 38 into base 6?

A. 26 B.

201 **C.**

102 D.

123

E. None of the above

21. Translate the three values below from their indicated current bases into base 10, and then identify the one whose value is different from the others (or select "None of the above" if all three base 10 values are identical).

A. 111 (base 8)

B. 141 (base 6)

C. 331 (base 4)

D. None of the above (all three values are identical)

111 base 8 = 64 + 8 + 1 = 73 base 10

*141 base 6 = 36 + 4*6 + 1 = 61 base 10*

*331 base 4 = 3*16 + 3*4 + 1 = 61 base 10*

22. The code below prints out a pattern of asterisks. How many asterisks in all will be printed to the screen?

```
for x in range(6):
    for y in range(x+1):
        print("*", end="")
    print()
```

A. 5

B. 6

C. 21

D. 36

E. None of the above

23. A student wrote the roomArea() function below, in order to collect the dimensions of a rectangular room, calculate its area, and return the answer. **BRIEFLY** identify **ALL** of the errors (in terms of both Python syntax and overall program logic) that you can find in this function.

```
def roomArea ():
    area = length ** width
    length = input("Please enter the length of the room in feet: ")
    return length
    width == input("Please enter the width of the room in feet: ")
    return area
```

- 'length' and 'width' should be read in before calculating the area
- The line that reads in 'length' should use int() to convert the input into an integer value
- The line that reads in 'width' should use int() to convert the input into an integer value
- The line that reads in 'width' uses the comparison operator (==) instead of assignment (=)
- The area calculation uses exponentiation (**) instead of multiplication (*)
- The function returns the value of 'length' (improperly) before it returns the area of the room

24. Consider the following Python function definition and determine what value will be returned by the function call that follows.

```
def calculate(value, src):
    result = 0
    val_string = str(value)
    size = len(val_string)
    mult = src ** (size - 1)

    for digit in val_string:
        d = int(digit)
        result += d * mult
        mult = mult // src

    return result

calculate(93, 2)
```

21

Quick (summarized) explanation:

1. 'size' is set to 2 (the number of digits in 'value')
2. 'mult' is set to 2 (the value of 'src'), raised to the first power ('size' - 1)
3. In the loop, 'd' is first set to 9. We add 9*2, or 18, to 'result'. Then 'mult' is divided by 2 using floor division, giving us 1.
4. The second round of the loop sets 'd' to 3. We add 3*1, or 3, to 'result', giving us a final answer of 21.

25. Write the decimal (base 10) equivalent of the hexadecimal (base 16) value 2B. Be sure to show your work in order to receive partial credit!

Solution 1: *$2B = (2 * 16^1) + (11 * 16^0) = 32 + 11 = 43$.*

Solution 2: *2B is equivalent to 00101011 in binary. 00101011 in binary is equal to 32 + 8 + 2 + 1, or 43.*

26. Complete the Python encode() function below, which takes a string as input and returns a new string (or "error") according to the following process:

1. Start with d = 1 and s = "".
2. Process the input string one character at a time. If you encounter a '*' character, multiply d by 2. If you encounter any other character, append it to the end of s and subtract 1 from d.
3. As soon as d reaches 0, return s.
4. If you run out of input characters (i.e., reach the end of the string argument) before d reaches 0, return the string "error".

```
def encode (input):
    d = 1
    s = ""

    for character in input:
        if character == "*":
            d = d * 2
        else:
            s = s + character
            d = d - 1

        if d == 0:
            return s

    return "error"
```

27. Consider a geometric sequence defined by the terms

$$a, ar, ar^2, ar^3, \ldots, ar^{n-1}$$

The first term of the sequence (meaning n is 1) is just the value of a. The second term (n is 2) is a times r, and so on. Complete the nthTerm() function below, which takes three integer arguments (a, r, and n) and returns the value of the nth term of this sequence.

```
def nthTerm (a, r, n):
    if n == 1:
        return a
    else:
        return a * (r ** (n-1))
```

28. Complete the Python function below, which recognizes (matches) strings that belong to a special lan- guage. The rules of this language are as follows:

- All strings in this language contain only the letters *a* and *b*
- A string only belongs to this language if it has equal numbers of *a*s and *b*s
- At any given point in the string, the number of *b*s that have been encountered must be less than or equal to the number of *a*s seen so far

For example, *aabb* is a valid string in this language, but *abba* is not (once the third character has been read, we have seen more *b*s than *a*s).

```
# Returns True if input belongs to the language, and False otherwise
# You may assume that input only contains 'a' and 'b' characters

def isInLanguage (input):
    num_as = 0
    num_bs = 0

    for letter in input:
        if letter == "a":
            num_as = num_as + 1
        else:
            num_bs = num_bs + 1

        if num_bs > num_as:
            return False # b's outnumber a's, so we abort processing

    if num_as == num_bs:
        return True # equal numbers of a's and b's
    else:
        return False
```