Question 1.
How does the Python interpreter parenthesize the following expression? Assume that a , b , c , d and e are variables of type ***int*** that have already been defined.
(a) a + (((b / c) ** d) * e)
(b) a + ((b / (c ** d)) * e)
(c) (a + b / c) ** (d * e)
(d) All of the above

Question 2.
E1 and E2 are boolean expressions. Consider the following expression:

not(E1 or E2) == (not E1 and not E2)

What can you say about the value of the expression given above?

(a) It is True if and only if E1 and E2 have different values
(b) It is False if and only if E1 and E2 have the same value
(c) It is always True
(d) It is always False

Question 3.
Consider the following statement:

word = 'abracadabra'

For what values of a and b does the following expression evaluate to True ? Assume that both a and b are integers. [MSQ]

word[a : b] == 'acad'

(a) a = 3, b = 7
(b) a = 4, b = 8
(c) a = -4, b = -8
(d) a = -8, b = -4

Question 4.
What will be the datatype of the following expressions? Choose the option that correctly matches
Expression to Data Type.

Expression	Datatype
(1) 15 % 5 // 5 * 10 ** 5	(A) `str`
(2) "1 + 2 + 3"	(B) `bool`
(3) 10 ** 10 / 5 + 25 - 50	(C) `float`
(4) 9 // 3 + 3 ** 3 > 3 * 3 * 3	(D) `int`
(5) 50 * 100.0 // 10 % 5	

.(a) 1-(B), 2-(C), 3-(A), 4-(D), 5-(C)
(b) 1-(A), 2-(D), 3-(C), 4-(B), 5-(C)
(c) 1-(D), 2-(A), 3-(C), 4-(B), 5-(C)
(d) 1-(C), 2-(A), 3-(D), 4-(B), 5-(D)
(e) 1-(C), 2-(A), 3-(C), 4-(B), 5-(C)

Question 5.
Consider the following code snippet:

```
a, b, c, d = input()
# Hint: input() always returns a string
d = 3
print((a + b + c) * d)
```

What will be the output of the code given above for the following input ?

Input: 1234

Select the correct **Output** from the options below:
(a) 123123123
(b) 1234
(c) 24
(d) 492
(e) 123412341234

Question 6: What will be the output of the code snippet given below?

```
L = [-1, 1]
for i in range(8):
    size = len(L)
    value = L[size - 2] + L[size - 1]
    L.append(value)
print(L)
```

(a) [-1, 1, 0, 1, 1, 2, 3, 5, 8, 13]
(b) [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
(c) [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
(d) [-1, -1, -2, -3, -5, -8, -13, -21, -34, -55]

Question 7.
For what values of a , b and c does the code given below print a sequence which has 0 as one of the elements? [MSQ]

```
# Hint: range(1, 9, 2) is the sequence: 1, 3, 5, 7
for i in range(a, b, c):
    print(i)
```

(a) a = 10, b = -1, c = -1
(b) a = -10, b = 1, c = 1
(c) a = 10, b = -2, c = 0
(d) a = 10, b = -2, c = 1

Question 8
Consider the following snippet of code:

```
word = input()
# Hint: word.isalpha() returns True only if
# every character in word is an alphabet
if word.isalpha():
    # Hint: word.lower() returns a string where
    # each alphabet character is in lower case
    word = word.lower()
    word = word + '12345'
else:
    word = 'abcd' + word
print(word)
```

When the code given above is executed, it prints the following ***output:*** abcd12345.

What would have been the input given by the user?[MSQ]
(a) abcd
(b) ABCD
(c) 12345
(d) 54321
(e) abcd12345
(f) dcba

Question 9.

In the following code-snippet, limit is an integer. For what value of limit does this code print 3.14 as output.

```
pi = '3.1415926535'
output = '' # there is no space between the quotes
index = 0
# limit is an integer
while len(output) < limit:
    output = output + pi[index]
    index = index + 1
print(output)
```

(a) 2
(b) 3
(c) 4
(d) 5

Question 10.
If n is a positive integer, what is the output of the following code? Assume that natural numbers start from 1, that is, 0 is not a natural number.

```
a = 0
# Hint: range(4, 8) is the sequence: 4, 5, 6, 7
for i in range(1, n + 1):
    b = 1
    for j in range(1, i + 1):
        b = b * j
    a = a + b
print(a)
```

(a) sum of the first n natural numbers
(b) product of the first n natural numbers
(c) sum of the factorial of the first n natural numbers
(d) factorial of the sum of the first n natural numbers

Question 11.
What is the output of the following snippet of code?

```
L = [[1,2,3], ['a', 'b', 'c'], 'abc']
count = 0
for elem in L:
    count = count + len(elem)
print(count)
```

(a) 3
(b) 4
(c) 6
(d) 9

Question 12.
What does the following code-snippet print?

```
def f(L, m):
    # Hint: L * m replicates the list m times
    # For example: [1, 2] * 3 will be [1, 2, 1, 2, 1, 2]
    L = L * m
    return L
x = [0]
y = 2
print(type(f(x, y)))
print(f(x, y))
```

Question 13.

What will be the output of the following snippet?

```
def myprint(msg, num):
    output = ''     # there is no space between the quotes
    for i in range(num):
        output = output + msg
    return output
print(myprint('python', 3))
```

Question 14.

Consider the following equation:

$$3x - 4y + 11z = 0$$

L is a non-empty list of lists. Each element of L is of the form [a, b, c] , where a , b and c are all integers. An element [a, b, c] of the list is a solution of the above equation if the following equation is satisfied:

$$3a - 4b + 11c = 0$$

Write a function is_plane that accepts this list L as input. It should return True only if every element in L is a solution of the equation and False otherwise. In other words, is_plane should return False even if a single element in it is not a solution. Select all correct implementations of this function. [MSQ]

(a)

```
def is_plane(L):
    for elem in L:
        a, b, c = elem[0], elem[1], elem[2]
        if 3 * a - 4 * b + 11 * c == 0:
            return True
        else:
            return False
```

(b)

```
def is_plane(L):
    for elem in L:
        a, b, c = elem[0], elem[1], elem[2]
        if 3 * a - 4 * b + 11 * c != 0:
            return False
    return True
```

(c)

```
def is_plane(L):
    for elem in L:
        count = 0
        a, b, c = elem[0], elem[1], elem[2]
        if 3 * a - 4 * b + 11 * c == 0:
            count = count + 1
    if count == len(L):
        return True
    return False
```

(d)

```
def is_plane(L):
    count = 0
    for elem in L:
        a, b, c = elem[0], elem[1], elem[2]
        if 3 * a - 4 * b + 11 * c == 0:
            count = count + 1
    if count == len(L):
        return True
    return False
```