Question 12 (2 points)

Complete the recursive function, getMatrixInfo, whose behavior is described in its docstrings.

```
def getMatrixinfo(mat):
    ''' (matrix (or list of list) of int) --> tuple( int, bool)
    Precondition mat is not empty and is a square matrix.
    Returns a tuple with 2 data, the first is the sum of the
    diagonal values, the 2nd is a boolean that is True if the
    matrix is a diagonal one or False otherwise.

    >>> getMatrixinfo([[1, 3, 0], [2, 4, 5], [4, 0, 8]])
    (13, False)
    >>> getMatrixinfo([[24, 0, 0], [0, 0, 0],[0, 0, 0]])
    (24, True)
    '''
```

Question 1 (1 point)

What does the following function achieve?

```
def fct(L,K) :
        if len(L) == 0 :
                return True
        return L[0] < K and fct(L[1:], K)
```

- Keeps looping for ever.
- returns True if the first element of L is smaller than K otherwise False.
- returns True if at least one of the elements of L is smaller than K otherwise False.
- None of the answers.
- returns True if all elements of L are smaller than K or False otherwise.

Question 2 (1 point)

Complete the 2 lines of code that start with # (remove the #), such that The fct correctly solves the problem described in the docstrings:

```
def fct(num):
    '''(int)->int
       Precondition: num is positive. The function
       computes 31 times num and returns that value
    '''
    # t =
    for i in range(num):
            # t =
    return t
```

Question 3 (1 point)

If the list *a* contains 10000 elements, how many times will line 3 be executed?

```
0 : def bubble_sort(a) :
1:      for i in range(len(a)) :
2:              for j in range(len(a) - 1) :
3:                      if a[j] > a[j+1] :
4 :                             a[j], a[j+1] = a[j+1], a[j]
```

Question 4 (1 point)

The following function is used on an integer list that is not empty, what does it return?

```
def test(list) :
            list.sort(list)
            if len(list)== 1:
                    return True

            for index in range(len(list)):
                    if index == len(list)-1 and list[index] != list[index-1]:
                            return True
                    elif index == 0 and list[index] != list[index+1]:
                            return True
                    elif index != 0 and index != len(list) - 1:
                            if list[index] != list[index-1] and list[index] != list[index+1]:
                                    return True
            return False
```

- None of the answers
- returns True if all elements of L are different, otherwise False.
- returns False if all elements of L are different, otherwise True.

Question 4 (1 point)

The following function is used on an integer list that is not empty, what does it return?

```
def test(list) :
        list.sort(list)
        if len(list)== 1:
                return True

        for index in range(len(list)):
                if index == len(list)-1 and list[index] != list[index-1]:
                        return True
                elif index == 0 and list[index] != list[index+1]:
                        return True
                elif index != 0 and index != len(list) - 1:
                        if list[index] != list[index-1] and list[index] != list[index+1]:
                                return True
        return False
```

- None of the answers
- returns True if all elements of L are different, otherwise False.
- returns False if all elements of L are different, otherwise True.
- returns False if L has at least 1 element that appears only once in L, otherwise True
- returns True if L has at least 1 element that appears only once in L, otherwise False.

Question 5 (1 point)

What does the following function achieve?

```
def fct(x):
        While len(x) > 3 :
                x.remove(min(x))
```

- Removes len(a)-4 smallest values from its list
- Keeps 4 of its largest values
- Keeps 3 of its largest values
- Keeps 3 of its smallest values
- None of the responses

Question 6 (1 point)

Consider the following function:

```
def fct(a, b):
        if a%b == 0 :
                return b
        else:
                return fct(b, a%b)
```

What doe the print instruction below display?

>>> print(fct(44, 12))

Question 7 (2 points)

What does the following piece of code display?

```
class Corner:

    def __init__(self, x =0, y =0):
        '''(Corner, int, int, str)-> None'''
        self.x = x
        self.y = y


    def __repr__(self):
      '''(Corner)-> str'''
      return "Corner(" + str(self.x) + "," + str(self.y) + ")"


def f(x, a, b):
    a = b
    b.x = 5
    a.y = 20
    x = x * 2
```

```
>>> x = 10

>>> co1 = Corner(2,6)

>>> co2 = Corner(5,10)

>>> f(x, co1, co2)

>>> print(x, co1, co2)
```

Question 8 (1 point)

Complete the #missing code (2nd line) in the following piece of code:

```
for i in range(1,7) :
        for j in #missing statement
                if j <= i :
                        print(j, end=" ")
                else:
                        print(" ", end=" ")
        print()
```

such that we can display the following:

```
        1
      2 1
    3 2 1
  4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
```

Question 9 (2 points)

Complete the recursive function, isPalindrome, whose behavior is described in its docstrings.

```
def isPalindrome(str):
    ''' str --> Bool
    Precondition: len(str)>0 and has only lower case letters
    Returns True if str is a palindrome (characters from
    left to right are the same as those from right to left.

    >>> str = 'abcba'
    >>> isPalindrome(str)
    True
    >>> isPalindrome("57899875")
    True
    >>> isPalindrome('xyz')
```

Question 10 (2 points)

Consider the following function:

```
def test(str, char):
        if len(str)-1 == 0:
                return str
        elif str[len(str)-1] == char:
                return test(str[:len(str)-1], char)
        else:
                return str[len(str)-1] + test(str[:len(str)-1], char)
```

What does the following instruction display?

```
>>> print(test('Popato Chef', 'p'))
```

Question 11 (1 point)

What does the following function achieve?

```
def fct(x) : ''' (list --> bool; Precondition: elements of x are numbers and x has
             at least 2 elements '''
        result = True
        for i in range(len(x) - 1) :
                if x[i] > x[i+1] :
                        result = False
                else :
                        result = True
        return result
```

- returns True if numbers of x are in increasing order otherwise False.

Question 11 (1 point)

What does the following function achieve?

```
def fct(x) : ''' (list --> bool; Precondition: elements of x are numbers and x has
             at least 2 elements '''
        result = True
        for i in range(len(x) - 1) :
                if x[i] > x[i+1] :
                        result = False
                else :
                        result = True
        return result
```

- returns True if numbers of x are in increasing order otherwise False.
- returns True if the last numbers in x is smaller than the one before otherwise False.
- returns True if numbers of x are in decreasing otherwise False.
- None of the answers
- returns True if the last numbers in x is larger than the one before otherwise False.

Question 12 (2 points)

Complete the recursive function, getMatrixInfo, whose behavior is described in its docstrings.

```
def getMatrixinfo(mat):
    ''' (matrix (or list of list) of int) --> tuple( int, bool)
    Precondition mat is not empty and is a square matrix.
    Returns a tuple with 2 data, the first is the sum of the
    diagonal values, the 2nd is a boolean that is True if the
    matrix is a diagonal one or False otherwise.
```

Question 13 (1 point)

Consider a list of a 1000 data ordered in an increasing order. We are looking for a particular x in it and decide to use a binary search approach. What will be the maximum number of steps that it will take to find x?

Question 14 (1 point)

Given a non-empty list of integers and the following function:

```
def fct(list):
    if len(list) == 0:
        return True
    return list[0] > 0 and fct(list[1:])
```

Complete the following statement:

```
# It returns True if ...
```

Question 15 (1 point)

What fragment of code is missing in the following function?

```
def fct(s1, s2) :
    '''(str, str) -> str
    returns a new chain with s1 characters that are at least once in s2.
    The characters in the result keep their order of occurrences in s1.

    >>> fct('abb', 'ab')
    'abb'
    >>> fct('abracadabra', 'ra')
    'araaara'
    '''

    res = ''
    # Missing fragment code.
    return res
```

Question 16 (1 point)

Implement a recursive function, recSumPosList(list), that takes a list as its only parameter. the function should return the sum of the positive elements of the input list.

```
>>> list = [1,-3,4,0,5]

>>> recSumPosList(list)

10
```

Add a File | Record Audio | Record Video

Question 17 (1 point)

If nb is a 3-digit integer, what instruction, that uses the minimum number of operators and parantheses, can provide the digit in the middle?

Question 18 (1 point)

Consider th following function:

```
def f_rec(a, left, right):
    if (right - left >= 1):
            f_rec( a, left+1, right-1)
            print(a[left], a[right], end = ' ')
```

>>> s = "abcdef"

What will be the outcome of the following call?

>>> f_rec(s, 0, len(s)-1)

Question 19 (1 point)

When does the following piece of code display 'Mount Kilimanjaro'?

```
def get():
    a = None
    try:
        a=float(input("Enter something:").strip())
    except:
        print("Mount Kilimanjaro")
    return a
```

Question 19 (1 point)

When does the following piece of code display 'Mount Kilimanjaro'?

```
def get():
    a = None
    try:
        a=float(input("Enter something:").strip())
    except:
        print("Mount Kilimanjaro")
    return a
```

- Never
- None of the answers.
- If the user enters a number with extra space after the number.
- If the user enters something that does not look like a number.
- Always

Question 20 (1 point)

A class Chose is defined using the method test:

```
Class Chose(object) :
        def test(self, a) :
                ….
```

If ***x*** is a variable that refers to an object of the class Chose and ***y*** is another variable, initialized to some value, how do we call the method test?

Question 6 (1 point) ✓ *Saved*

Consider the following function:

```
def fct(a, b):
        if a%b == 0 :
                return b
        else:
                return fct(b, a%b)
```

What doe the print instruction below display?

>>> print(fct(44, 12))

4

```
0 : def bubble_sort(a) :
1:      for i in range(len(a)) :
2:              for j in range(len(a) - 1) :
3:                      if a[j] > a[j+1] :
4 :                             a[j], a[j+1] = a[j+1], a[j]
```

Question 1 (1 point)

What does the following function achieve?

```
def fct(L,K) :
        if len(L) == 0 :
                return True
        return L[0] < K and fct(L[1:], K)
```

- Keeps looping for ever.
- returns True if the first element of L is smaller than K otherwise False.
- returns True if at least one of the elements of L is smaller than K otherwise False.
- None of the answers.
- returns True if all elements of L are smaller than K or False otherwise.

Question 8 (1 point)

Complete the #missing code (2nd line) in the following piece of code:

```
for i in range(1,7) :
        for j in #missing statement
                if j <= i :
                        print(j, end=" ")
                else:
                        print(" ", end=" ")
        print()
```

such that we can display the following:

```
        1
      2 1
    3 2 1
  4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
```

Question 5 (1 point) ✓ *Saved*

What does the following function achieve?

```
def fct(x):
        While len(x) > 3 :
                x.remove(min(x))
```

- (●) Removes len(a)-4 smallest values from its list
- ( ) Keeps 4 of its largest values
- ( ) Keeps 3 of its largest values
- ( ) Keeps 3 of its smallest values
- ( ) None of the responses