Name: _____ Date: _____

Pledge: _____

Total: ___ / 75

Closed book: no textbook, no electronic devices, one sheet of paper with notes. Read each question carefully before answering! You may work out the solution to each question within the test itself, but **only your answers on this cover page will be graded**.

Question 1

a) _____7_____ (3 points)

b) _____['B', 'X']_____ (3 points)

c) _____['C', 'D']_____ (4 points)

Question 2

_____b, d_____ (10 points)

Question 3

a) _____3_____ (5 points)

b) _____Bad_____ (5 points)

c) _____0.0_____ (5 points)

Question 4

a) _____16_____ (4 points)

b) ____i < abs(n)?___ (3 points)

c) _____i += 1_____ (3 points)

Question 5

a) _____n_____ (5 points)

b) _____i_____ (5 points)

c) _____n - i - 1_____ (5 points)

Question 6

a) _____len(lst)_____ (5 points)

b) _____lst[i] < i_____ (5 points)

c) ___total += lst[i]___ (5 points)

Question 1 (10 points)
Consider the following code in each section. Assume each section is independent. What is printed on the screen?

```
a) L = [1, 2, [3, 4, 5], 6]
   M = list(L)
   M[2][2] = 7

   print(L[2][2])

b) L = ['A', ['B', ['C', 'D']]]
   M = L
   M[1][1] = 'X'

   print(L[1])

c) L = ['A', ['B', ['C', 'D']]]
   M = deepcopy(L)
   M[1][0] = 'X'
   M[1][0] = 'y'

   print(L[1][1])
```

Question 2 (10 points)
Consider the following code:

```
def search(lst, target):
    first = 0
    last = len(lst) - 1
    while first <= last:
        middle = first + (last - first) // 2
        if target == lst[middle]:
            return middle
        if target < lst[middle]:
            last = middle - 1
        else:
            first = middle + 1
    return -first - 1
```

Assume `lst` is `[3, 8, 9, 14, 18, 33, 65, 87, 99]`.
For which values of `target` will `search` return -4? Select one or more answers.

(a) 18
(b) 10
(c) -4
(d) 13
(e) None of the above

Question 3 (15 points)
Consider the following code in each section. Assume each section is independent. What is printed on the screen?

a)
```
lst = [1, 2, 3]
try:
    a = lst[2 % 3]
except ValueError:
    print('Bad')
    sys.exit(1)
print(a)
sys.exit(0)
```

b)
```
try:
    a = int('X15')
    print(a)
    sys.exit(0)
except ValueError:
    print('Bad')
    sys.exit(1)
```

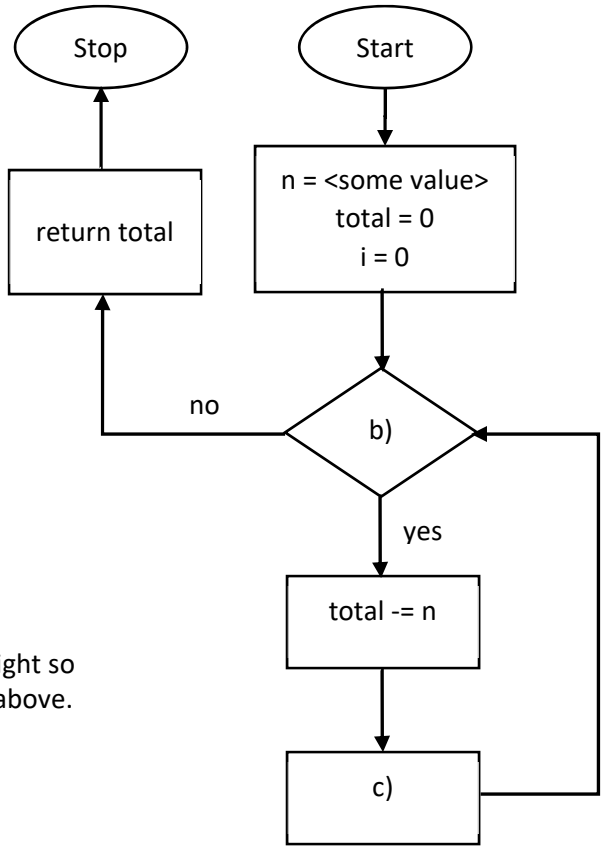
c)
```
x = 5
y = 0
try:
    print(x / y)
except:
    print(y / x)
    sys.exit(1)
sys.exit(0)
```

Question 4 (10 points)

```
def confuse(n):
    total = 0
    for i in range(abs(n)):
        total -= n
    return total
```

a) What value is returned by `confuse(-4)`?

b) – c) Fill in the symbols in the flowchart to the right so that the logic is identical to that of the Python code above.



Question 5 (15 points)
The function `sum_min_diag` should add all the values in the minor diagonal of a matrix of any size and return that sum.

The `data` parameter is a 2D list, or more specifically, a list of lists. `data` will have an equal number of rows and columns, i.e. it has dimension n x n. For example, a 4 x 4 matrix would be defined as follows:

```
data = [[1, 2, 3, 4],
        [5, 6, 7, 8],
        [9, 10, 11, 12],
        [13, 14, 15, 16]]
```

The minor diagonal starts in the bottom left and goes toward the top right. In the example below, elements in the minor diagonal have been shaded. The sum of the values in the minor diagonal is 34.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Fill in the blanks to complete this function.

```
def sum_min_diag(data):
    total = 0

    n = len(data)

    for i in range(0, ____________________, 1):                # a)

        total += data[____________________][____________________] # b) and c)

    return total
```

Question 6 (15 points)
Fill in the blanks to complete the following function.

```
def sum_values_less_index(lst):
    '''Assume lst is a list of integers.
    Return the sum of only values that are less than the value of the index
    in which they reside.

    If lst is empty, return 0.

    For example,
    sum_values_less_index([]) should return 0.
    sum_values_less_index([-5, -1, 42, 3]) should return -6.
    -5 is less than 0, so it is part of the sum.
    -1 is less than 1, so it is part of the sum.
    Nothing else is added.
    sum_values_less_index([-2, 1, 1, 1, 2]) should return 2.
    -2 is less than 0, so it is part of the sum.
     1 is not less than 1, so it is not part of the sum.
     1, 1, and 2 are less than indices 2, 3, and 4, respectively, and are
    therefore part of the sum.
    '''

    total = 0

    for i in range(0, _________________, 1):      # a)

        if _________________:                     # b)

            _________________                     # c)

    return total
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Question 7 (15 points)
' Implement missing sections of the Employee class.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
class Employee(object):
    '''Write the constructor below. It should take in five arguments:
       - first_name (a string)
       - last_name (a string)
       - title (a string)
       - hours_per_week (an int)
       - hourly_rate (a float)
       All fields must be private. No error checking or type conversions
       are required.
       5 points'''
    def __init__(self, first_name, last_name, title, hours_per_week, \
                 hourly_rate):
        self.__first_name = first_name
        self.__last_name = last_name
        self.__title = title
        self.__hours_per_week = hours_per_week
        self.__hourly_rate = hourly_rate
```

```
    '''Write a property for hourly_rate. 3 points'''
    @property
    def hourly_rate(self):
        return self.__hourly_rate

    '''Write a setter for hourly rate. 3 points'''
    @hourly_rate.setter
    def hourly_rate(self, hourly_rate):
        self.__hourly_rate = hourly_rate

    '''Write a method called get_total_compensation.
       It returns the total amount of money an employee earns in a year.
       Assume that the employee works 50 weeks each year, with the remaining
       2 set aside for vacation.
       4 points'''
    def get_total_compensation(self):
        return 50 * self.__hours_per_week * self.__hourly_rate

    def __str__(self):
        return 'Employee: %s %s\n  Title: %s\n  Hours per week: %d\n' \
               '  Hourly rate: $%.2f\n  Yearly compensation: $%.2f' % \
            (self.__first_name, self.__last_name, self.__title, \
             self.__hours_per_week, self.__hourly_rate, \
             self.get_total_compensation())

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Question 8 (15 points)
' Implement missing sections of the Manager class. Manager should be a
' subclass of Employee.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
class Manager(Employee):  # 2 points
    '''Write the constructor below. It should take in six arguments:
    - the first five are the same as in the Employee constructor
    - bonus_percent, a float >= 0. This attribute represents the percentage
      of the employee's yearly compensation that will be used to
      create the manager's annual bonus. MAKE SURE the argument is a float
      >= 0.  Otherwise, if it's not a float raise a TypeError stating,
      "Bonus percent must be a float." If it's a float but < 0, raise a
      ValueError stating, "Bonus percent cannot be negative."
      bonus_percent must be private.
    8 points'''
    def __init__(self, first_name, last_name, title, hours_per_week, \
                 hourly_rate, bonus_percent):
        super().__init__(\
            first_name, last_name, title, hours_per_week, hourly_rate)
        try:
            self.__bonus_percent = float(bonus_percent)
        except:
            raise TypeError("Bonus percent must be a float.")
        if self.__bonus_percent < 0:
            raise ValueError("Bonus percent cannot be negative.")
```

```
'''Override the method get_total_compensation.
It returns the total amount of money the manager earns in a year, i.e.
basic employee compensation + bonus.
To get full credit, you must call get_total_compensation in the superclass.
Note: If a manager's yearly compensation is $100,000 and the bonus_percent
      is 10 (ten), the total compensation will be 110,000.
5 points'''
def get_total_compensation(self):
    return (1 + self.__bonus_percent / 100) * \
           super().get_total_compensation()
```