Name: _____ Date: _____

Pledge: _____

Total: ___ / 75

Closed book: no textbook, no electronic devices, one sheet of paper with notes. Read each question carefully before answering, as there is **no partial credit for any response**! 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) [4, 6] (3 points)

b) [4, 5] (3 points)

c) [1, 2] (4 points)

Question 2

a) e. 7 (5 points)

b) b. target not found (5 points)

Question 3

a) 3.14 (5 points)

b) Whoops (5 points)

c) Bad (5 points)

Question 4

a) a. b. e. or 3, 5, 13 (4 points)

b) is d < n? (2 points)

c) is n % d == 0? (2 points)

d) d += 1 (2 points)

Question 5

a) n (3 points)

b) n (3 points)

c) row (3 points)

d) col (3 points)

e) sums (3 points)

Question 6

a) [] (3 points)

b) len(lst) (3 points)

c) lst[i] (3 points)

d) lst[max_index] (3 points)

e) i (3 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] ]
M = L
M[2][1] = 6

print(L[2])
```

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

print(L[2])
```

c)
```
L = [ [1, 2], 3, [4, 5] ]
M = deepcopy(L)
M[0][0] = 0
M[0][1] = 0

print(L[0])
```

Question 2 (10 points)
Consider the following code:

```
def search(lst, target):
    n = len(lst)
    for i in range(n // 2 + 1):
        if lst[i] == target:
            return i
        if lst[n - i - 1] == target:
            return n - i - 1
    return -1
```

Assume `lst` is `[0, 2, 4, 5, 1, 9, 3, 4, 7]`.

a) What will `search(lst, 4)` return?
   a) 3
   b) 2
   c) -1
   d) 8
   e) 7
   f) None of the above

b) What input causes the search function to make the most comparisons?
   a) Every number in the list is the same as the target
   b) The target is not found in the list
   c) There are duplicates of the target in the list
   d) Every number in the list is unique

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

a)
```
try:
    a = float('3.14')
    print(a)
    sys.exit(0)
except ValueError:
    print('Bad')
    sys.exit(1)
except TypeError:
    print('Really Bad')
    sys.exit(1)
```

b)
```
lst = [5, 3]
try:
    a = lst[5 % 3]
except ValueError:
    print('Bad')
    sys.exit(1)
except IndexError:
    print('Whoops')
    sys.exit(1)
print(a)
sys.exit(0)
```

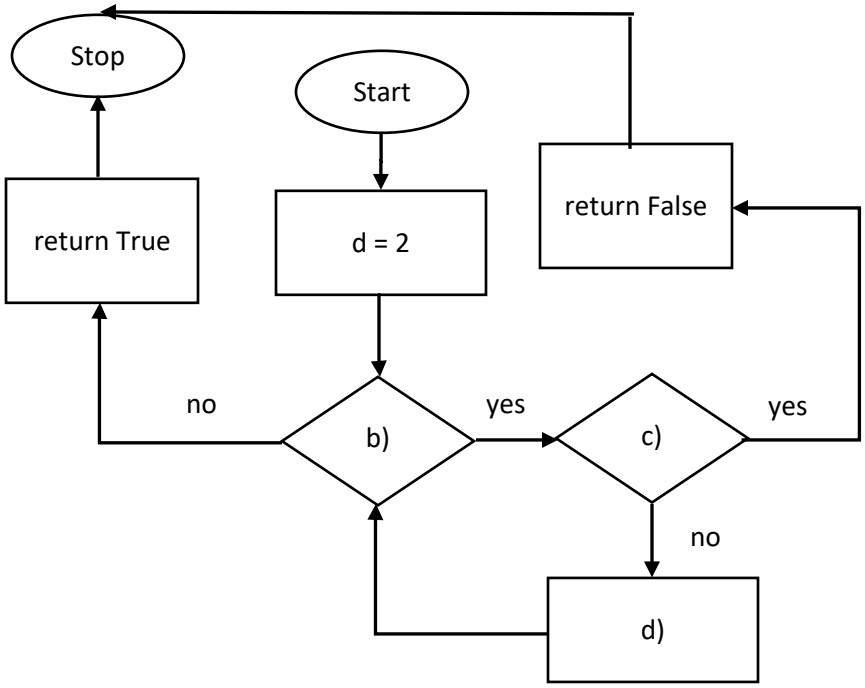
c)
```
x = 0
y = 5
try:
    print(y % x)
except:
    print('Bad')
    sys.exit(1)
sys.exit(0)
```

Question 4 (10 points)
Consider the following code:

```
def mystery(n):
    d = 2
    while d < n:
        if n % d == 0:
            return False
        d += 1
    return True
```

a) For which values of n does `mystery(n)` return True? (More than one answer may be correct. List all that apply.)
   a) 3
   b) 5
   c) 8
   d) 10
   e) 13

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



Question 5 (15 points)
The function `max_column_sum` should add all the values in each of the columns and return the maximum sum found among all the columns.

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

In this example, the four columns sum to [21, 2, 35, 21]. You'll want to return 35, the maximum sum found across all the columns.

Here is the matrix:

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

Here are the column sums:

21	2	35	21

35, shaded, is the max.

Fill in the blanks to complete this function. Do not use the len() function in any of your answers.

```
def max_column_sum(data):
    n = len(data)

    sums = [0] * n

    for col in range(0, ________(a)_________, 1):

        for row in range(0, ________(b)_________, 1):

            sums[col] += data[________(c)_________][________(d)_________]

    return max(_______(e)_________)
```

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

```
def index_of_max(lst):
    '''Assume lst is a list of integers.
    Return the index of the maximum value in lst. You MAY NOT call Python's
    built-in max() function.

    If lst is empty, return -1.
    For example,
    index_of_max([]) should return -1.
    index_of_max([5, 1, 42, 3]) should return 2.
    index_of_max([1, -3, 8, 9, 2]) should return 3.
    '''
    if lst == _____(a)_____:

        return -1

    max_index = 0

    for i in range(1, _____(b)_____, 1):

        if _____(c)_____ > _____(d)_____:

            max_index = _____(e)_____

    return max_index
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Question 7 (15 points)
' Implement missing sections of the MotorBoat class.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
class MotorBoat(object):
    '''Write the constructor below. It should take in four arguments:
       - overall_length (a float representing the total length in meters)
       - waterline_length (a float representing the length in meters where
                           the boat meets the water)
       - weight (an int representing the weight of the boat in pounds)
       - horsepower (an int representing the horsepower of the motor)
       All fields must be private. No error checking or type conversions
       are required.
       5 points'''
    def __init__(self, overall_length, waterline_length, weight, horsepower):
        self.__overall_length = overall_length
        self.__waterline_length = waterline_length
        self.__weight = weight
        self.__horsepower = horsepower

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

```
    '''Write a setter for horsepower. 3 points'''
    @horsepower.setter
    def horsepower(self, horsepower):
        self.__horsepower = horsepower

    '''Write a method called get_max_speed.
       It returns the maximum speed of the boat based on the horsepower of
       the motor and the total weight of the boat.
       The formula is: 225 x sqrt(horsepower / weight)
       4 points'''
    def get_max_speed(self):
        return 225 * sqrt(self.__horsepower / self.__weight)

    def __str__(self):
        return '%s:\n  Overall length: %.1f meters\n' \
               '  Waterline length: %.1f meters\n' \
               '  Weight: %d pounds\n  Horsepower: %d\n' \
               '  Max speed: %.1f mph' % \
            (self.__class__.__name__, self.__overall_length, \
             self.__waterline_length, self.__weight, \
             self.__horsepower, self.get_max_speed())



''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Question 8 (15 points)
' Implement missing sections of the MotorSailor class. MotorSailor should be
' a subclass of MotorBoat.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
class MotorSailor(MotorBoat):  # 2 points
    '''Write the constructor below. It should take in five arguments:
    - the first four are the same as in the MotorBoat constructor.
    - sail_dimension, a float >= 0. This attribute represents the length
      and width in meters of the square sail attached to the boat.

      MAKE SURE sail_dimension is a float >= 0. Otherwise, if it's not a float
      raise a TypeError stating, "Sail dimension must be a float."  You must
      use the type() function to get credit for this part.
      If it's a float but < 0, raise a ValueError stating,
      "Sail dimension cannot be negative."

      sail_dimension must be private.

    8 points'''
    def __init__(self, overall_length, waterline_length, weight, horsepower,
                 sail_dimension):
        super().__init__(overall_length, waterline_length, weight, horsepower)

        if type(sail_dimension) != float:
            raise TypeError("Sail dimension must be a float.")
        if sail_dimension < 0:
            raise ValueError("Sail dimension cannot be negative.")
        self.__sail_dimension = sail_dimension
```

```
'''Override the method get_max_speed.
It returns the maximum speed the boat can travel.
The maximum speed is the sum of speed obtained from the motor plus
the boost picked up from the sail.
The formula for sail boost is the sail's area in square meters / 3.5.
To get full credit, you must call get_max_speed in the superclass.
5 points'''
def get_max_speed(self):
    return super().get_max_speed() + self.__sail_dimension ** 2 / 3.5
```