CS 115 Exam 3, Fall 2015, Sections 5-8

Your name: _____

Rules

- You may use one handwritten 8.5 x 11" cheat sheet (front and back). This is the only resource you may consult during this exam.
- Explain/show work if you want to receive partial credit for wrong answers.
- As long as your code is correct, you will get full credit. No points for style.
- When you write code, be sure that the indentation level of each statement is clear.

	Your Score	**Max Score**
Problem 1: Binary search		10
Problem 2: Selection sort		10
Problem 3: Mergesort		10
Problem 4: Recursion		20
Problem 5: Defining classes		25
Problem 6: Using classes		25
Total		**100**

Reference code for Problems 1 and 2

The functions below are just for your reference on Problems 1 and 2. You do not need to read them if you understand the algorithms.

```
def binary_search(search_list, value_to_find):
      first = 0
      last = len(search_list) - 1

      while first <= last:
            middle = (first + last) // 2
            # Problem 1: state the values of first, last,
            # and middle at this point in the code
            if value_to_find == search_list[middle]:
                  return middle
            elif value_to_find < search_list[middle]:
                 last = middle - 1
            else:
                  first = middle + 1
```

```
def selection_sort(list_to_sort):
  for i in range(len(list_to_sort) - 1):
    min_index = find_min_index(list_to_sort, i)
    list_to_sort[i], list_to_sort[min_index] =
               list_to_sort[min_index], list_to_sort[i]
    # Problem 2: Show list contents at this point


def find_min_index(L, s):
     min_index = s
     for i  in range(s, len(L)):
        if L[i] < L[min_index]:
            min_index = i
    return min_index
```

Reference code for Problem 3

The functions below are just for your reference on Problem 3. You do not need to read them if you understand the algorithms.

```
def merge(L, start_index, sublist_size):
    index_left = start_index
    left_stop_index = start_index + sublist_size
    index_right = start_index + sublist_size
    right_stop_index = min(start_index + 2 * sublist_size,
                           len(L))
    L_tmp = []

    while (index_left < left_stop_index and
           index_right < right_stop_index):
        if L[index_left] < L[index_right]:
           L_tmp.append(L[index_left])
           index_left += 1
        else:
           L_tmp.append(L[index_right])
           index_right += 1

    if index_left < left_stop_index:
           L_tmp.extend(L[index_left : left_stop_index])
    if index_right < right_stop_index:
           L_tmp.extend(L[index_right : right_stop_index])

    L[start_index : right_stop_index] = L_tmp

def merge_sort(L):
    chunksize = 1
    while chunksize < len(L):
        left_start_index = 0  # Start of left chunk in each pair
        while left_start_index + chunksize < len(L):
            merge(L, left_start_index, chunksize)
            left_start_index += 2 * chunksize

        chunksize *= 2
        # Problem 3: Show list contents at this point
```

Problem 1: Binary search (10 points)

Consider the following sorted list:

```
L = [   'adele',
        'beiber',
        'bryan',
        'crane',
        'jovi',
        'roxette',
        'santana',
        'swift' ]
```

and the binary search code on page 2. You may want to label the elements of L with their numeric index values before proceeding.

(a) Fill out the following table tracing the call `binary_search(L, 'santana')`, a binary search for `'santana'` in this list, according to the comment in the code. ***You should fill out one row per iteration of the loop***. If there are more rows than iterations, leave the extra rows blank.

Iteration	Value of `first`	Value of `last`	Value of `middle`	Value of `L[middle]`
1				
2				
3				
4				
5				

(b) Fill out the following table tracing call to `binary_search(L, 'boyzone')`, a binary search for `'boyzone'` in this list.

Iteration	Value of `first`	Value of `last`	Value of `middle`	Value of `L[middle]`
1				
2				
3				
4				
5				

Problem 2: Selection sort (10 points)

Consider the following list:

```
L = [   'swift',
        'jovi',
        'crane',
        'beiber',
        'roxette',
        'adele',
        'santana',
        'bryan']
```

In the diagrams below, show the contents of the list after each of the first 4 iterations of the for-loop in `selection_sort`. If the list does not change from one iteration to the next, you can write "SAME" for the next iteration.

INDEX	INITIAL ORDER	AFTER i=0 ITERATION	AFTER i=1	AFTER i=2	AFTER i=3
0	swift				
1	jovi				
2	crane				
3	beiber				
4	roxette				
5	adele				
6	santana				
7	bryan				

Problem 3: Mergesort (10 points)

Consider the following list:

```
L = [   'swift',
        'jovi',
        'crane',
        'beiber',
        'roxette',
        'adele',
        'santana',
        'bryan']
```

In the diagrams below, show the contents of the list after each of the first 3 iterations of the outer while-loop in `merge_sort`. If the list does not change from one iteration to the next, you can write "SAME" for the next iteration.

INDEX	INITIAL ORDER	AFTER chunksize=1 ITERATION	AFTER chunksize=2 ITERATION	AFTER chunksize=4 ITERATION
0	swift			
1	jovi			
2	crane			
3	beiber			
4	roxette			
5	adele			
6	santana			
7	bryan			

Problem 4a: Recursion (15 points)

Consider the following function definition:

```
def rec(L,e):
    # parameter L is a list of numbers
    # parameter e is a number
    if len(L) == 0:
        return True
    if L[0] > e:
        return False
    return rec(L[1:],e)
```

A. What does the following snippet of code return?

```
L = [2]
rec(L, 0)
```

B. Show the chain of recursive calls, and state what the return value is for each call, starting with:

```
L = [27, 16, 36, 4, 9]
rec(L, 36)
```

C. How would you summarize what this function does in one sentence? Don't explain the code line-by-line. Provide a higher-level description like "adds *x* and *y*" or "computes *x* factorial."

Problem 4b: Recursion (5 points)

Consider the following function definition:

```
def A(x) :
     print(x)

     if (x == 0) :
        return 1
     else :
        r = x//2
        return 1 + A(r)
```

Specify the output (from the print statement) and the return value `val` obtained with the following function call: `val = A(5)`

Output:

Return value `val`:

Problem 5: Defining classes (25 points)

In this problem, you will define a class to represent a `Pirate`.

If you use the input() or print() functions in your solution to this problem, you're doing it wrong!

Your class should be named `Pirate`, and you should define the following methods:

`__init__`: This method initializes a `Pirate` object. Initialize the attributes to store the pirate's name (eg. 'Pegleg Pete'), how many gold doubloons he has (eg. 144), a boolean value to indicate whether he has a hookhand or not.

`__str__`: This method returns a string with the `Pirate` object's attributes, formatted as follows:

```
Pirate Pegleg Pete has 144 gold doubloons. Ahoy! He
does not have a hookhand.
```

or

```
Pirate Hookhand Harry has 95 gold doubloons. Ahoy! He
has a hookhand.
```

The above output is just an example: you should use the actual values in place of values that are underlined. Note the second statement is based on whether the Pirate has a hookhand or not.

`__lt__`: This method compares `self` to another `Pirate` object.
It returns `True` if the `self` object has less number of gold doubloons than another `Pirate` object, and `False` otherwise

`get_gold_count`: This method returns the amount of gold doubloons

`is_hookh`: This methods returns `True` if the pirate has a hookhand and `False` otherwise

`repair`: For the pirates who have a hookhand, this method repairs it so that they do not have a hookhand anymore. In addition, it gives them 5 extra gold doubloons.

`rob`: This method takes in another `Pirate` object and takes away the doubloons belonging to it and gives them to the current object (`self`).

[Write code in next page]

[WRITE YOUR PROBLEM 5 CODE HERE]

The last page of this exam has extra space for you to write your solution.

Problem 6: Using classes (25 points)

For this problem, you must write a **complete program**. However, you can assume that the `Pirate` class from Problem 5 has already been correctly defined for you.

To earn full credit, you must use the methods of the `Pirate` class whenever possible.

Read the instructions carefully before you start coding!

Your program should do the following:

1. A function called `CreatePirate` to do the following:
   - Ask the user to enter the pirate name (consisting of exactly two words) and gold doubloons on the same line. For example:
     `Pegleg Pete 144`
   - If the user's line is blank, **return** None
   - Otherwise, if the user did not provide exactly 3 values separated by whitespace, exit the program with an error message.
   - If the number of gold doubloons is not a POSITIVE number, exit the program with an error message.
   - Otherwise, if one of the words in pirate's name is hookhand (in any combination of uppercase or lowercase letters), then this pirate has a hookhand. For example, if pirate's name is "`HookHand Harry`", he has a hookhand.
   - Create and **return** a `Pirate` object that uses the information the user entered.

2. A main function to do the following:
   - Call `CreatePirate` repeatedly until the user enters a blank line instead of the pirate information.
   - Use the results of `CreatePirate` to build a list of pirates and print out each pirate's information consisting of name, gold doubloons and whether he has a hookhand or not.
   - Using your `Pirate` methods, find
     - Total number of pirates with hookhand and print it.
     - The pirate with least number of gold doubloons. Let's call him PirateA.
     - Any pirate with a hookhand (assume at least one exists). Let's call him PirateB.
     - Have PirateA rob PirateB.
     - PirateB is upset, so repair his hookhand and print out his new information.

The last page of this exam has extra space for you to write your solution.

[WRITE YOUR PROBLEM 6 CODE HERE]

[EXTRA SPACE FOR PROBLEMS 5 AND 6]

[EXTRA SPACE FOR PROBLEMS 5 AND 6]