CS 115 Exam 3, Spring 2016, 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 = [   'circle',
        'exit',
        'feedback',
        'goal',
        'guess',
        'mastermind',
        'palette',
        'state' ]
```

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 `v=binary_search(L, 'guess')`, a binary search for `'guess'` 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. At the end, write the value `v` returned by the function

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

Return value `v`:

(b) Fill out the following table tracing call to `v=binary_search(L, 'click')`, a binary search for `'click'` in this list. At the end, write the value `v` returned by the function

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

Return value `v`:

Problem 2: Selection sort (10 points)

Consider the following list:

```
L = [   'exit',
        'goal',
        'state',
        'guess',
        'circle',
        'palette',
        'feedback',
        'mastermind']
```

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	exit				
1	goal				
2	state				
3	guess				
4	circle				
5	palette				
6	feedback				
7	mastermind				

Problem 3: Mergesort (10 points)

Consider the following list:

```
L = [   'goal',
        'palette',
        'guess',
        'exit',
        'state',
        'circle',
        'mastermind',
        'feedback']
```

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	goal			
1	palette			
2	guess			
3	exit			
4	state			
5	circle			
6	mastermind			
7	feedback			

Problem 4a: Recursion (15 points)

Consider the following function definition:

```
def fun(n, a):
    # parameter n is an integer
    # parameter a is an integer

    if n <= 0:
        return 1
    else:
        return a * fun(n-1, a)
```

A. What does the following snippet of code return?

```
fun(0, 2)
```

```
fun(1, 2)
```

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

```
fun(5, 2)
```

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 func(n) :
    # parameter n is an integer

    if n<=0:
       print(0)
    else:
       print(n%10, end="")
       func(n//10)
```

Specify the output (from the print statement) obtained with the following function call: `func(253)`

Output:

Problem 5: Defining classes (25 points)

In this problem, you will define a class to represent a `Bank Account`.Your class should be named `Account`, and you should define the following methods:

`__init__`: This method initializes an `Account` object. Initialize the attributes to store the account holder's name (eg. Mark Perry), balance in the account (eg. 50) and whether the account is checking or savings.

`get_name`: This method returns the account holder's name

`get_balance`: This method returns the current balance in the account

`is_checking`: This methods returns `True` if the account is `checking` and `False` if it is `savings`

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

`Mark Perry holds a checking account for the amount of $50`

The above output is just an example: you should use the actual values in place of values that are underlined.

`__lt__`: This method compares `self` to another `Account` object.
It returns `True` if the `self` object has less balance than another `Account` object, and `False` otherwise

`deposit`: This methods deposits a certain amount to the current `Account` object (`self`), where the amount is a parameter to the method

`withdraw`: This methods withdraws a certain amount from the current `Account` object (`self`), where the amount is a parameter to the method, and returns a boolean indicating the success/failure of withdrawal. That is, if the amount being withdrawn is less than or equal to the current balance it returns `True`. Otherwise, the withdrawal is canceled, an error message is displayed to the user and the method returns `False`

`transfer`: This method transfers $50 from the current `Account` into another `Account`. Use the `withdraw` and `deposit` methods to accomplish this transfer.

[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 `Account` class from Problem 5 has already been correctly defined for you.

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

Read the instructions carefully before you start coding!

Your program should do the following:

1. A function called `CreateAccount` to do the following:
   - Ask the user to enter the account holder's name and initial deposit on two lines. For example:
     ```
     Mark Perry
     50
     ```
   - If the user's name is `Darth Vader` (characters can be in lower or upper case), **return** None
   - If the amount for deposit is negative, reset it to `0`.
   - If the initial deposit is more than `500`, it will be a `savings` account; otherwise it will be a `checking` account.
   - Create and **return** an `Account` object that uses the information the user entered.

2. A main function to do the following:
   - Call `CreateAccount` repeatedly until the user's name is `Darth Vader`.
   - Use the results of `CreateAccount` to build a list of accounts and print out each account's information consisting of name, account balance and whether it is a checkings or savings account.
   - After creating the list, use the methods of the `Account` class to find
     - Total number of savings account and print it.
     - The account with the least balance. Let's call it *minAcc*.
     - Any account belonging to "Luke Skywalker" (you can assume one exists). Let's call it *lukeAcc*.
   - Now, transfer all money from *lukeAcc* to *minAcc*.
   - Luke is unhappy. Deposit $15 into his account and print his new account 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]