CS 115 Exam 3 (Section 2) Spring 2018 Tu. 05/15/2018

Name: ____

Rules and Hints

- You may use one handwritten 8.5 x 11" cheat sheet (front and back). This is the only additional resource you may consult during this exam. *No calculators.*
- When you write code, be sure that the indentation level of each statement is clear.
- 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.
- As always, the SSU rules on academic integrity are in effect.

Problem	**Max Score**	**Your Score**
Problem 1: Binary Search	10	
Problem 2: Selection Sort	10	
Problem 3: Mergesort	10	
Problem 4: Object-Oriented Analysis	15	
Problem 5: Defining classes	25	
Problem 6: Using classes	30	
Total	100	

Cheat Sheet Additions

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
    return None

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

def swap(L, i, j):
    x = L[i]
    L[i] = L[j]
    L[j] = x

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
```

Cheat Sheet Additions

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 = ['double-razorback',
     'double-trotter',
     'jowler',
     'leaning-jowler',
     'mixed-combo',
     'pig-out',
     'razorback',
     'sider',
     'snouter',
     'trotter']
```

Problem 1A Fill out the below table, tracing `v = binary_search(L, 'leaning-jowler')`, a binary search for `'leaning-jowler'` in this list. Fill out one row per iteration of the loop (per the comment in the code on page 2). If there are more rows than iterations, leave extra rows blank. At the end, write the function's return value `v`.

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

Return value `v`: _________

Problem 1B Fill out the below table, tracing the call `v = binary_search(L, 'pig-flys')`, a binary search for `'pig-flys'` in this list. Fill out one row per iteration of the loop (per the comment in the code on page 2). If there are more rows than iterations, leave extra rows blank. At the end, write the function's return value `v`.

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

Return value `v`: _________

Problem 2: **Selection Sort (10 points)**

Consider the following list:

```
L = ['stone',
     'hallows',
     'azkaban',
     'goblet',
     'prince',
     'chamber',
     'pheonix']
```

In the table below, show the *contents* of the list after each of the first four iterations of the for-loop in `selection_sort` (per the comment in the code on page 2).

You may just draw a horizontal line between cells if a word has *not* changed position.

Index	Initial Order	After $i = 0$ iteration	After $i = 1$ iteration	After $i = 2$ iteration	After $i = 3$ iteration
0	stone				
1	hallows				
2	azkaban				
3	goblet				
4	prince				
5	chamber				
6	pheonix				

Problem 3: **Mergesort (10 points)**

Consider the following list:

```
L = ['lucy',
     'rerun',
     'frieda',
     'eudora',
     'shermy',
     'sally',
     'franklin',
     'marcie']
```

In the diagrams below, show the contents of the list after each of the first three iterations of the outer while-loop in `merge_sort` (per the comment in the code on page 3).

Index	Initial Order	After `chunksize == 1`	After `chunksize == 2`	After `chunksize == 4`
0	lucy			
1	rerun			
2	frieda			
3	eudora			
4	shermy			
5	sally			
6	franklin			
7	marcie			

Problem 4: **Object-Oriented Analysis (15 points)**

Answer the questions below, using the following Python code:

```
class Account:                          # 4E:
    def __init__(self, b):
        self.bal = b

    def __str__(self):
        return str(self.bal)

    def add(self, b):
        self.bal += b

a = 11                                  # 4D: Calls method ___________

b = Account(a)                          # 4D: Calls method ___________

b.add(a)                                # 4D: Calls method ___________

print(a)                                # 4D: Calls method ___________

print(b)                                # 4D: Calls method ___________
```

Problem 4A What is the data type of the variable `a`? ________________________

Problem 4B What is the data type of the variable `b`? ________________________

Problem 4C List the methods of class `Account`.

Problem 4D In each comment labeled 4D above, fill in the blank with the method(s) of class `Account` that are called in the execution of that line. If a line does not call a method of class `Account`, write N/A.

Problem 4E In the box labeled 4E above, write the output of the code.

Problem 5: **Defining classes (25 points)**

In this problem, you will define a class to represent a news article. Your class should be named `NewsArticle`, and you should define the methods below. Hint: if you are using the `print` or `input` functions to implement these methods, you are doing it wrong.

`__init__`: Initializes a `NewsArticle` object. Takes two parameters: name of the publisher (a string) and number of words in the article (an integer) and saves these in appropriate attributes.

In addition, it defines attributes to

- store a list of hashtags associated with article and
- store a Boolean value indicating if the article is fake or genuine.

Initially, there are no hashtags associated with article (list is empty) and article is considered genuine.

`is_fake`: Returns `True` if the article is fake, and `False` if the article is genuine.

`get_words`: Returns the number of the words in the article.

`get_publisher`: Returns the name of the publisher of the article.

`add_tag`: Takes one parameter—a hashtag (string) associated with the article—and adds it to the list of hashtags.

`update_fake`: Takes one parameter—a string—and if this string is found in the current list of hashtags (irrespective of case of letters), the article is labeled to be fake. For example, if string = `'#SNL'` and list of hashtags = `['#satire', '#snl', '#storm']`, then article is fake.

`__lt__`: Compares `self` to another `NewsArticle` object. It returns `True` if `self` has a smaller number of words than the other object, and `False` otherwise.

`__str__`: Returns a string summarizing the `NewsArticle` object, following the format below exactly (two examples follow):

```
Article in Washington Post with 3 hashtags and 2319 words
is fake!
```

or :

```
Article in Politico with 17 hashtags and 925 words is
genuine.
```

You should use values derived from attributes in place of the underlined values.

Start your solution on the next page...
Toward the end of the exam, there are extra pages if needed.

Problem 5, continued . . .

Problem 6: **Using classes (30 points)**

For this problem, you must write a complete program. This includes logic in `def main()`, a call to `main()`, any necessary library imports, etc. You do *not* need to write any docstrings.

To earn full credit, you must *use the methods* from the `NewsArticle` class whenever appropriate. You may assume that the class, as described in Problem 5, has already been correctly implemented for you. Read the instructions carefully before you start coding!

Your program should do the following:

1. Define a function called `CreateArticle` that does the following:
   - Prompt for the length of the article. If length is 0, return `None`.

     ```
     Number of words in your article: 2319
     ```
   - Prompt for the publisher of the article.

     ```
     Publisher: Washington Post
     ```
   - Prompt the user for how many hashtags they will associate with the article.

     ```
     Number of hashtags: 3
     ```
   - Prompt for each hashtag in the following format:

     ```
     Hashtag 1: #Satire
     Hashtag 2: #snl
     Hashtag 3: #storM
     ```
   - Create a `NewsArticle` object, using all the data that has been entered above.
   - Return the `NewsArticle` object created.
2. Define a function called `NewsToday` that creates and *returns* a list of articles by:
   - Calling `CreateArticle` repeatedly, until it returns `None`.
   - Each time `CreateArticle` returns a `NewsArticle` object, it inserts it into a list.
3. Define a function called `main` that does the following:
   - Call `NewsToday` to get a list of `NewsArticle` objects.
   - Prompt user for a hashtag that is indicative of fake news.
   - For any `NewsArticle` object associated with this hashtag, update the status (fake/genuine) of the object to label it *fake*.
   - Print information about each *genuine* article (as provided by the `__str__` method).
   - Among all the *fake* articles, determine the largest article (as provided by the `__lt__` method) and print its information.

Start your solution on the next page...
Toward the end of the exam, there are extra pages if needed.

Problem 6, continued . . .

Extra Pages . . .