CS 115 Exam 3, Spring 2014

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.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Your Score</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1: Binary search</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Problem 2: Selection sort</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Problem 3: Recursion</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Problem 4: 2D lists</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Problem 5: Defining classes</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>Problem 6: Using classes</td>
<td></td>
<td>25</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>
Reference code for Problems 1 and 2

The 3 functions below are just for your reference on Problems 1 and 2. You do not need to read them if you understand the algorithms.

```python
# binary_search()
# Finds the position of an item in a list
# Parameters: the list; the item to search for
# Returns: the item’s position (or None)
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)
        list_to_sort[i], list_to_sort[min_index] =
        # 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
```

CS 115 Exam 3, Spring 2014
Page 2 of 10
Problem 1: Binary search (10 points)

Consider the following sorted list:
\[
L = ['apple',
     'banana',
     'carrot',
     'durian',
     'eggplant',
     'fava',
     'grape',
     'jicama',
     'kumquat',
     'lychee']
\]

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 a binary search for 'carrot' 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.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Value of first</th>
<th>Value of last</th>
<th>Value of middle</th>
<th>Value of (L[\text{middle}])</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Fill out the following table tracing a binary search for 'honeydew' in this list.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Value of first</th>
<th>Value of last</th>
<th>Value of middle</th>
<th>Value of (L[\text{middle}])</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 2: Selection sort (10 points)
Consider the following list:
\[
L = ['students', 'and', 'profs', 'totally', 'love', 'early', 'morning', 'exams']
\]

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.

<table>
<thead>
<tr>
<th>INDEX</th>
<th>INITIAL ORDER</th>
<th>AFTER i=0 ITERATION</th>
<th>AFTER i=1</th>
<th>AFTER i=2</th>
<th>AFTER i=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>students</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>and</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>profs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>totally</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>love</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>early</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>morning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>exams</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 3: Recursion (10 points)

Consider the following function definition:

```python
def magic(s1):
    # parameter is a string or list
    if len(s1) == 0:
        return 0
    if s1[0].lower() == 'z':
        return 1 + magic(s1[1:])
    return magic(s1[1:])
```

A. What does the following function call return?
   ```python
   L = []
   magic(L)
   ```

B. Show the chain of recursive calls, and state what the final return value is for the call:
   ```python
   magic('sizzle')
   ```

C. How would you summarize what this function does in just a few words?
   Don't explain the code line-by-line. Provide a higher-level description like
   "adds x and y" or "computes x factorial."
Problem 4: 2D lists (15 points)

For this problem, assume that L is a 2D list and that every element of L is the same length (i.e., L contains the same number of rows and columns).

(a) Finish this function definition, as specified:

```python
def column_check(L, num):
    # Assumes that L is a 2D list of numbers.
    # Returns True if each column of L adds up to a value less than num.
    # Returns False if one or more columns of L add up to a value greater than num.
```

(b) Finish this function definition, as specified:

```python
def count_X(L):
    # Returns the number of times 'X' or 'x' appears as an element of the 2D list L.
```
Problem 5: Creating classes (30 points)
In this problem, you will define a class to represent a dieter’s daily caloric intake.

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

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

__init__: This method initializes a Dieter object.
  • Parameter: the dieter’s target number of calories for the day
  • Initializes: the dieter’s target number of calories and the number of calories the dieter has consumed.

__str__: This method returns a string with the dieter’s target number of calories and the number they have left to consume, formatted exactly as follows:

| Target: 1400; Consumed: 1200; Remaining: 200 |
| or |
| Target: 1400; Consumed: 1500; Excess: 100 |

reset: Resets the number of calories to 0. Doesn’t return anything.

add_meal: Takes the number of calories in a meal as a parameter and adds it to the number consumed. Doesn’t return anything.

add_exercise: Takes the calories burned as a parameter and subtracts it from the number consumed. Doesn’t return anything.

remaining: Returns the number of calories the dieter has left to consume; can be negative if the dieter has consumed more calories than the target amount.

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 Dieter class from Problem 5 has already been correctly defined for you.

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

Read the instructions carefully before you start coding!

Your program should do the following. You can assume that the user enters non-negative integer inputs.

- Ask the user how many dieters live in their household.
- Prompt the user for each dieter’s calorie target, and create a Dieter object for each person.
- Ask the user how many meals they ate today. You can assume that everyone in the household ate exactly the same things.
- Prompt the user for the number of calories in each meal.
- Ask the user how many calories they burned today.
- Print an updated report for each dieter. For example:

Dieter 1:
Target: 1400; Consumed: 1200; Remaining: 200

Dieter 2:
Target: 1500; Consumed: 1200; Remaining: 300

The last page of this exam has extra space for you to write your solution.
[EXTRA SPACE FOR PROBLEMS 5 AND 6]