NAME (print): ________________________________
Netid: __________________
Community Standard Acknowledgement (signature): ________________________________

Do NOT spend too much time on any one question.

In writing code you do not need to worry about specifying the proper import statements. Assume that all libraries we’ve discussed are imported in any code you write.

Do not discuss this test with anyone until the test is returned. Do not use the web, PyCharm, or any programming environment that checks your code.

<table>
<thead>
<tr>
<th>Problem</th>
<th>value</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>2 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>30 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>30 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>18 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>80 pts.</td>
<td></td>
</tr>
</tbody>
</table>
PROBLEM 1:  (*Honor code (2 pts)*)

Print your name to acknowledge the Duke Community Standard
Print your name again to say you have read all the rules for the exam and agree to follow them (including not using the web, classmates, Pycharm, or other environments to determine answers to your program).

PROBLEM 2:  (*Short code segments (30 pts)*)

For each of the following problems, use only what is indicated to set result to a Python expression. Unless specified within the problem, DO NOT use any Python methods or simply add a string that solves the problem.

*Here is an example.*
Use phrase with indexing and the concatenation of two items to set result to the string ’by’.

```python
phrase = 'bicycle'
result = phrase[0] + phrase[3]
```

Note this answer uses only phrase and indexing, and the concatenation of two items. It does not simply assign the string `result = 'by'`

**PART A (3 pts)**

Use phrase with splicing and concatenation of two items to create the string ’tryher’.

```python
phrase = 'thermometry'
result = ______________________
```

**PART B (3 pts)**

Use phrase with indexing and concatenation of three items to create the string mod.

```python
phrase = ['doberman']
result = ______________________
```

**PART C (3 pts)**

Use lst with indexing and concatenation to create the string ’top’.

```python
lst = ['computer']
result = ______________________
```
PART D (3 pts)
Use lst with indexing and the concatenation of two items to make the string 'fond'.

```python
lst = [['first', 'second', 'third'], 'fourth']
result = ________________
```

PART E (3 pts)
Use lst with indexing to create ['one', 'two'].

```python
lst = [['one', 'two'], 'three', ['four']]  
result = ________________
```

PART F (3 pts)
Use only lst with slicing to create the string 'Green'.

```python
lst = [['Durham', 'Greensboro'], ['Charlotte'], 'Raleigh']
result = ________________
```

PART G (3 pts)
Using slicing only, create a clone of lst

```python
lst = ['pear', ['plum', 10], 'apple']
result = ________________
```

PART H (3 pts)
Using the minimal slicing and concatenation, create the string "Hove".

```python
phrase = 'Houston we have a problem.'
result = ________________
```

PART I (3 pts)
Using the minimal indexing and concatenation, create '823'.

```python
lst = ['55', '24', '8', '3', '61']
result = ________________
```

PART J (3 pts)
Using the minimal indexing and concatenation, create 'Fall2021'.

```python
lst = ['Winter', 2020, 'Spring', '2021', 'fall', 2019, 'Fall']
result = ________________
```
PART A: List Creation (15 pts)

Create the function `compareString` that has two parameters: `text1` and `text2`. The function first determines which is the larger string, then returns a list that contains a new string (the concatenation of `text1` and `text2`, with the larger string first) and the total length of the new, combined string. If the strings are equal, then the function should still return a list containing the concatenated string and the correct length.

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>compareString('Help', 'Halle')</code></td>
<td>['HelpHalle', 9]</td>
</tr>
<tr>
<td><code>compareString('active', 'Active')</code></td>
<td>['activeActive', 12]</td>
</tr>
<tr>
<td><code>compareString('HELP', 'HOLE')</code></td>
<td>['HOLEHELP', 8]</td>
</tr>
<tr>
<td><code>compareString('HELP', 'HELP')</code></td>
<td>['HELPHELP', 8]</td>
</tr>
</tbody>
</table>

Complete the function `newString` below.

```python
def compareString(text1, text2):
```
A movie theater sells tickets at the following prices based on age, as noted below.

<table>
<thead>
<tr>
<th>Age</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 and under(toddler)</td>
<td>FREE</td>
</tr>
<tr>
<td>ages 7-12(child)</td>
<td>$5</td>
</tr>
<tr>
<td>ages 13-54(adult)</td>
<td>$10</td>
</tr>
<tr>
<td>ages 55 and up(senior)</td>
<td>$5</td>
</tr>
</tbody>
</table>

Create the function `purchase` that has four parameters: `toddler`, `child`, `adult`, and `senior`, which correspond to the total number of tickets purchased for each type. The function should then calculate and output to the screen the total number of tickets sold that day (including toddler tickets) and the total ticket price as "Quantity:X Price:$Y", where X is the total tickets and Y is the total price.

<table>
<thead>
<tr>
<th>call</th>
<th>output</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>purchase(5, 2, 4, 0)</td>
<td>&quot;Quantity:11 Price:$50&quot;</td>
<td>$0 for 5 toddler tix, $10 for 2 child, and $40 for 4 adult.</td>
</tr>
<tr>
<td>purchase(5, 0, 2, 0)</td>
<td>&quot;Quantity:7 Price:$20&quot;</td>
<td>$0 for 5 toddler tix, $20 for the two adult.</td>
</tr>
<tr>
<td>purchase(1, 1, 5, 5)</td>
<td>&quot;Quantity:12 Price:$80&quot;</td>
<td>$0 for 1 toddler, $5 for 1 child, $50 for 5 adult, $50 for 5 senior.</td>
</tr>
</tbody>
</table>

Complete the function `purchase` below.

```python
def purchase(toddler, child, adult, senior):
```
PART A) (8 pts)
Consider the following program:

```python
list1 = ['Kim', 'Tori', ['Michele', 'Teri']]
if 'Teri' in list1:
    test = True
else:
    test = False
list2=list1[:]
list1[0]=list2[-1]
print(list1)
```

a) What is the expected output of the program?

b) Using only indexing and two additional lines of code, how would you update list1 to now store ['Tori', 'Tori', 'Tori']?
PART B) (10 pts)
Consider the following program:

```python
def compare(state1, state2):
    if len(state1) == len(state2):
        if state1 < state2:
            print(state1 + " should be listed first.")
        else:
            print(state2 + " should be listed first.")
    else:
        print(True)

if name1 == name2:
    print("The strings are identical!")
else:
    print("The strings are NOT identical!")

if __name__ == '__main__':
    name1 = 'North Carolina'
    name2 = 'South Carolina'
    name3 = 'Virginia'
    result = compare(name1, name3)
    print("The result of the comparison is", result)
```

This program doesn’t execute. This code contains at least one error.

a) What line(s) of code contains the error(s)?

b) What error(s) are present?

c) Rewrite the line(s) of code with the correct version only as 16: corrected code, where 16 is the line number of the code and corrected code is the rewritten line of code.