Name:_____ **Date:**____/____/_____

CSCSCI-UA.0002 – Midterm #2 – List Practice Questions

1. Write a program that prints out the index and the element value separated by an equal sign of every element in the following list (you don't have to create your own function to do this). You must use a for loop to do this.

```
food_emoji = ['Mushroom', 'Chestnut', 'Bread', 'Cheese Wedge', 'Meat on Bone']
```

The output should be:

```
0 = Mushroom
1 = Chestnut
2 = Bread
3 = Cheese Wedge
4 = Meat on Bone
```

```
for i in range(len(food_emoji)):
    print('{} = {}'.format(i, food_emoji[i]))
```

2. Using the same `food_emoji` as above, print out the last character of the last string using one line and indexing (the result is `'e'`):

```
print(food_emoji[-1][-1])
```

3. Using the same `food_emoji` list as the problems above, with one line of code, swap `Cheese Wedge` with `Mushroom` … the resulting list should be: `['Cheese Wedge', 'Chestnut', 'Bread', 'Mushroom', 'Meat on Bone']`

```
food_emoji[0], food_emoji[3] = food_emoji[3], food_emoji[0]
```

4. Using the altered `food_emoji` from the problems above, create a new list composed of all of the unicode code points (numeric values associated with a unicode character) of all of the characters in the first element of the list of food emoji names. Print this list out. The result will be (these are the unicode code points of every character in Cheese Wedge!):

```
[67, 104, 101, 101, 115, 101, 32, 87, 101, 100, 103, 101]
```

```
numbers = []
for c in food_emoji[0]:
    numbers.append(ord(c))
print(numbers)
```

5. Using the altered `food_emoji` list as the problems above, fill in the blanks for the following program so that it matches the output below (assume that food emoji is defined above). Notice that it stops once the last element is removed.

```
# keep on going as long as we have items in our list...

while _________len(food_emoji) > 1_________:

    print('All food: {}'.format(food_emoji))

    if input('Remove {}?\n> '.format(______food_emoji[-1]__________)) == 'Y':

        food_emoji.______pop()__________________
```

Expected Output

```
All food: ['Cheese Wedge', 'Chestnut', 'Bread', 'Mushroom', 'Meat on Bone']
Remove Meat on Bone?
> Y
All food: ['Cheese Wedge', 'Chestnut', 'Bread', 'Mushroom']
Remove Mushroom?
> N
All food: ['Cheese Wedge', 'Chestnut', 'Bread', 'Mushroom']
Remove Mushroom?
> Y
All food: ['Cheese Wedge', 'Chestnut', 'Bread']
Remove Bread?
> Y
All food: ['Cheese Wedge', 'Chestnut']
Remove Chestnut?
> Y
All food: ['Cheese Wedge']
Remove Cheese Wedge?
> Y
```

6. Create a function called **`create_student_report`** that will return a string representation of a list of student test scores. The list of student scores is composed of sub lists, with each sub list representing a student. The sub list will have the first name, last name and test score at the first, second and third index respectively. For example, this is a list of 4 students' names and scores:

```
gradebook = [['Alice', 'Ang', 70], ['Bob', 'Basara', 80], ['Carol', 'Castillo', 90]]
```

The resulting report (string representation of the names and scores) is last name, followed by comma... then first name follwed by colon... and finally the score followed by new line:

```
Ang, Alice: 70
Basara, Bob: 80
Castillo, Carol: 90
```

The function should meet the following specifications:

a) **parameters**: a list of student scores in the format above
b) **processing**: creates a string representation of the report using the format above
c) **return value**: the string reprsenting the report

Example Usage:

```
print(create_student_report(gradebook))
# Prints out...
# Ang, Alice: 70
# Basara, Bob: 80
```

```
def create_student_report(students):
    report = ''
    for s in students:
        report += '{}, {}: {}\n'.format(s[1], s[0], s[2])
    return report
```

7. Using the same format for a list of test scores as above, create a function called **`calculate_median`** that gives back the median test score of all of the scores passed in. The median is the middle of a list of values... for example, 4 is the median of 2, 4, 6. If there are two "middle" numbers, take the average of both numbers... for example, 5 is the median of 2, 4, 6, 8.

The function should meet the following specifications:

a) **parameters**: a list of student scores in the format above
b) **processing**: creates a string representation of the report using the format above
c) **return value**: the string reprsenting the report

Example Usage:

```
gradebook = [['Alice', 'Ang', 70], ['Bob', 'Basara', 80], ['Carol', 'Castillo', 90]]
print(calculate_median(gradebook))
# prints out 80

gradebook = [['Alice', 'Ang', 70], ['Bob', 'Basara', 80], ['Carol', 'Castillo', 90], ['Deborah',
'Diya', 100]]
print(calculate_median(gradebook))
# prints out 95.0
```

```
def calculate_median(students):
    # assume that students is not an empty list
    scores = []
    for student in students:
        scores.append(student[2])
    scores.sort()
    num_students = len(students)
    # remember that division converts to float, so wrap with call to int...
    if num_students % 2 == 1:
        return scores[int((num_students - 1) / 2)]
    else:
        # take the average of the middle 2 numbers
        return (scores[int(num_students / 2)] + scores[int(num_students / 2 + 1)]) / 2
```

8. Assume that you have a list of words that looks like this to start: `words = ['foo', 'bar', 'baz']`

   Show three ways to use a method to add the element 'qux' to this list (position doesn't matter).

```
words.append('qux')
words.extend(['qux'])
words.insert(0, 'qux')

# you can use an index that's doesn't exist to insert at the end:
words.insert(100, 'qux')
```

9. Write a function called **`filter_plurals`** that takes a list of words as an argument and returns a new list of words composed of all words that are **not** plural (singular) from the original list. Naively assume that if a word ends in s, it's a plural. Example usage:

```
print(filter_plurals(['cats', 'dog', 'book', 'chairs']))
# prints out ['dog', 'book']

def filter_plurals(words):
    new_words = []
    for word in words:
        if word[-1].lower() != 's':
            new_words.append(word)
    return new_words
```

10. Create a function called **`avg_positive_sum`** that gives back the average sum of every list of numbers in a list of lists of numbers that has a sum that's greater than 0. For example, the list, [[1, 1, 1], [-10, 2, 3], [10, 20], [3, 2, 2, 5]], has 4 sublists. The sums for each are 3, -5, 30, and 12. After discarding the list that has a sum of -5, the average of all of the sums is 9. Example usage:

```
print(avg_positive_sum([[1, 1, 1], [-10, 2, 3], [10, 20], [3, 2, 2, 5]]))

def avg_positive_sum(my_list):
    grand_total = 0
    # we have to keep track of count since we may not include all
    # sub lists...
    count = 0
    for numbers in my_list:
        total = 0
        for n in numbers:
            total += n
        if total > 0:
            count += 1
            grand_total += total
    return grand_total  / count
```

11. Continually ask the user for positive numbers. Once they enter a number that's 0 or less, ask them for a threshold (one last number). Print out a report with the following information (you can implement this without cre:

    a) a list representation of all of the numbers
    b) a list representation of all numbers greater than or equal to the threshold (if the threshold is greater than all numbers, then display an empty list)
    c) the average of all of the numbers greater than the threshold (if there's an empty list, print out a message saying that there are no numbers to find the average of)

    Example output:

```
Give me a number
> 9
Give me a number
> 1
Give me a number
> 7
Give me a number
> 3
Give me a number
> -1
Give me a threshold
> 5
All numbers [9, 1, 7, 3]
Numbers >= 5: [9, 7]
The average of numbers >= 5: 8.0
```

```
response = 1
numbers = []
while response > 0:
    response = int(input('Give me a number\n> '))
    if response > 0:
        numbers.append(response)
threshold = int(input('Give me a threshold\n> '))
new_numbers = []

# note that the loop bodies of the following two loops can
# be put into a single for loop - two for loops are used
# here for clarity

# create a new list with numbers >= threshold
for num in numbers:
    if num >= threshold:
        new_numbers.append(num)

# find the average
if len(new_numbers) > 0:
    total = 0
    for num in new_numbers:
        total += num
    avg = total / len(new_numbers)

print('All numbers {}'.format(numbers))
print('Numbers >= {}: {}'.format(threshold, new_numbers))
if len(new_numbers) > 0:
    print('The average of numbers >= {}: {}'.format(threshold, avg))
else:
    print('No numbers to find average of.')
```

```
# find the average
total = 0
for num in new_numbers:
    total += num
avg = total / len(new_numbers)

print('All numbers {}'.format(numbers))
print('Numbers larger than {}: {}'.format(threshold, new_numbers))
print('The average of numbers larger than {}: {}'.format(threshold, avg))
```

12. Create a list of random numbers, with the number of elements in the resulting list also random. Print out the list. Then, swap every other element with the next element in the list. Print out the list again.

a) The total number of random numbers can be 5 through 15, inclusive.
b) The possible numbers in the list can be 1 through 9, inclusive.
c) Print out "original", followed by the resulting list after creating the list of random numbers.
d) Go through every other element in the list, starting with the first.
e) Swap the element with the adjacent element (the element at the next index, without skipping)
f) For example, if the original list were `[1, 2, 3, 4]`, the new list would be `[2, 1, 4, 3]`
g) Print out "new list", followed by your modified list
h) Example output:

Run 1
```
original: [2, 1, 5, 3, 5, 1, 1, 6, 5, 5, 7, 2, 7, 1, 3]
new list: [1, 2, 3, 5, 1, 5, 6, 1, 5, 5, 2, 7, 1, 7, 3]
```

Run 2
```
original: [3, 7, 7, 8, 7, 3]
new list: [7, 3, 8, 7, 3, 7]
```

```
# create a list of random numbers (with a random length)
import random
my_list = []
for n in range(random.randint(5, 15)):
    my_list.append(random.randint(1, 9))
print('original:', my_list)

# swap every other element in the list
for idx in range(0, len(my_list), 2):
    # make sure there's a next element
    if idx < len(my_list) - 1:
        my_list[idx], my_list[idx + 1] = my_list[idx + 1], my_list[idx]
print('new list:', my_list)
```