Important

1. Due Date: **Thursday, September 28th**
2. This homework is graded out of 100 points.
3. This is an Individual Assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
    - TA Helpdesk (Schedule posted on class website.)
    - Email TA's or use Piazza Forums Notes:
    - How to Think Like a Computer Scientists [http://openbookproject.net/thinkcs/python/english3e/]
    - CS 1301 Python Debugging Guide [http://www.cc.gatech.edu/classes/AY2016/cs1301_spring/CS-1301-Debugging-Guide/index.html]
5. Don't forget to include the required collaboration statement (outlined on the syllabus). Failing to include the Collaboration Statement will result in no credit.
6. Do not wait until the last minute to do this assignment in case you run into problems.
7. Comment or delete all your function calls. When your code is run, all it should do is build without any errors. Having function calls or extraneous code outside the scope of functions will result in a lower grade. (import statements and global variables are okay to be outside the scope of a function)
8. **Read the entire specifications document before starting this assignment.**
9. **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%.**

Introduction

The goal of this homework is for you to showcase your knowledge about how you can use indexing to manipulate strings and lists. The functions you are asked to code will all use some sort of indexing to accomplish the goal. Refer to the rubric to see how points will be rewarded for each function. You have been given HW4.py to fill out with instructions in the docstrings. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin. You have until Thursday, September 28th to complete this assignment.

Function Name: **remove_duplicates**
Parameters: aList
Return value: The modified list.

Description: Write a function that takes in a list. Return a new list that has removed all of the duplicates from the first list. The modified list should have the unique elements in the same order as the original list. If there are no duplicates, then return the original list. If the original list is empty return an empty list.

Test Cases:

```
>>> newList = remove_duplicates([10, "cat", 5.2, "hello", "cat",
"Hello", 0])
>>> print(newList)
[10, 'cat', 5.2, 'hello', 'Hello', 0]

>>> newList = remove_duplicates([10, 35, 35, 10, 98, "hi", 35,
"hi", 0, 0])
>>> print(newList)
[10, 35, 98, 'hi', 0]
```

Function Name: **common_elements**
Parameters: list1 and list2
Return value: True if the two lists have at least one common element. False if they are different.

Description: Write a function that takes in two lists. If the two lists have at least one common element, return True. If the two lists are completely different, return False. You can assume that none of the lists will be empty.

Test Cases:

```
>>> returnValue = common_elements([1, 2, 3, 4], [6, 7, 8, 9, 10])
>>> print(returnValue)
False

>>> returnValue = common_elements(["goodbye", 88, True,
"yellow"], ["orange", False, "red", "goodbye", 0])
>>> print(returnValue)
True
```

Function Name: **`flatten_list`**
Parameters: list1
Return value: A list that contains the elements of the list passed in and the elements of the nested lists.

Description: Write a function that takes a list and returns a new list. If any of the elements in the list are also lists, the returned list should have the individual elements of that nested list. You can assume that the nested list will not contain any elements that are also lists. If the original list is empty you can return an empty list.

Hint: Try using the `type()` function.

Test Cases:

```
>>> newList = flatten_list([8, 9, [0, True, "ok"], False])
>>> print(newList)
[8, 9, 0, True, 'ok', False]

>>> newList = flatten_list(["hi", [5, "ok"], 8, "purple", [7,
"bye", False], True])
>>> print(newList)
['hi', 5, 'ok', 8, 'purple', 7, 'bye', False, True]
```

Function Name: **`make_odd_even`**
Parameters: list1
Return Value: The modified list.

Description: Write a function that takes a list and returns a new list. You can assume that the list passed in will only contain integers. If the elements of the list are even, add them to the new list. If the elements of the list are odd, make them even by adding 1 and then add that new number to the list. If the original list is empty you can return an empty list. You may assume the integer 0 is even.

Test Cases:

```
>>> newList = make_odd_even([2, 43, 23, 34, 20, 88, 7])
>>> print(newList)
[2, 44, 24, 34, 20, 88, 8]

>>> newList = make_odd_even([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
>>> print(newList)
[0, 2, 2, 4, 4, 6, 6, 8, 8, 10, 10]
```

Function Name: **unique_lists**
Parameters: list1, list2
Return Value: The modified list1.

Description: Write a function that takes in two lists. If an element in list2 is not in list1, append that element to list1. If an element in list2 is already in list1, remove that element from list1. Return the updated list1.

Test Cases:

```
>>> newList = unique_lists([6, "butterfly", "waffle", 19], [True,
"shirt", 8, "waffle", 6, "computer"])
>>> print(newList)
['butterfly', 19, True, 'shirt', 8, 'computer']

>>> newList = unique_lists(["water", "sand", "beach", 14],
[False, "earring", "smile", 17])
>>> print(newList)
['water', 'sand', 'beach', 14, False, 'earring', 'smile', 17]
```

Function Name: **process_string**
Parameters: *aStr, aNum* (int)
Return value: a list of ints

Description: Write a function that accepts a string of characters, filters out all non-number characters from the string, and returns a list of the numbers in the string that are a factor of *aNum*. NOTE: You can also ignore all zeroes (0) in the string, as 0 is not a factor of any number.

Test Cases:

```
>>> process_string("jahfig234thdajg0gri9t2832488radga", 12)
[2, 3, 4, 2, 3, 2, 4]

>>> process_string("aoweh102935jgalhfda03191jfwl", 7)
[1, 1, 1]

>>> process_string("!@#^%&*&$%@$^^$%$#%^&*(){}|<><,.,??./", 31)
[]
```

Function name: **`find_family`**
Parameters: *names_list* (a list of strings); *surname* (string)
Return value: a list containing the first names of all the people that have *surname* AND do not have middle names.

Description: Write a function that takes in a list of names and a specific surname to search for. Return a list of the UNIQUE first names of all the names in *names_list* that have *surname* AND do not have middle names. NOTE: Each name could have 0 or more middle names, but all names have a first and last name.

Test Cases:

```
>>> names_1 = ["Josh Washington", "Chris Hartley", "Sam
Giddings", "Beth Washington", "Jessica Riley", "Ashley Brown",
"Hannah Washington", "Mike Munroe", "Matt Taylor", "Emily Davis"]
>>> family_1 = find_family(names_1, “Washington”)
>>> print(family_1)
['Josh', 'Beth', 'Hannah']

>>> names_2 = ["Esteban Julio Ricardo Montoya de la Rosa
Ramirez", "Lina Ramirez", "Sebastian Alejandro Ramirez", "Zack
Martin", "Cody Martin", "Jonathan Romero", "Daniela Ramirez"]
>>> family_2 = find_family(names_2, "Ramirez")
>>> print(family_2)
['Lina', 'Daniela']

>>> names_3 = ["John Roberts", "Anthony Kennedy", "Clarence
Thomas", "Ruth Bader Ginsburg", "Stephen Breyer", "Samuel Alito",
"Sonia Sotomayor", "Elena Kagan", "Neil Gorsuch"]
>>> family_3 = find_family(names_3, "O'Connor")
>>> print(family_3)
[]
```

Function name: **`add_next`**
Parameters: *aList* (a list of integers)
Return value: None
Description: Overwrite each element in aList with its value added with the value of the next element in the list. The last element on the list (or if there is only one element in the list) should be added to itself.

Note: Be aware that we are asking you to change the list in place, that is we are **not asking you to return a new list** with the modified changes, we want you to override the list we provide as an argument. Read this for more info and help:
http://interactivepython.org/runestone/static/thinkcspy/Lists/UsingListsasParameters.html

Test Cases:

```
>>> things = [2, 5, 9]
>>> add_next(things)
>>> print(things)
[7, 14, 18]

>>> things2 = [3]
>>> add_next(things2)
>>> print(things2)
[6]
```

Function name: **`election_day`**
Parameters: a nested list; each list will contain three values: the first value will be the name of a candidate (str), the second value will be the total number of votes this candidate received (int) , and the third parameter will be the two-letter abbreviation of a U.S. state (str).
Return value: a string representation of the winning candidate, the total number of votes the candidate received, and the state in which the candidate received the most votes
Description: You are helping total the votes for a nationwide political election. Write a function that returns a string with the name of the winning candidate (i.e. the candidate who received the highest total number of votes across all states), the total number of votes for that candidate, and the state in which the candidate received the most votes. If two candidates are tied for highest number of votes, your function should return information for the candidate that appears first in the list.

Note: Failing to return the exact format of the string will cause you to lose points. (Format: Candidate name[comma][space] total # of votes [comma][space] two-letter state abbreviation)

Test Cases:

```
>>> election_1 = [["Pepsi", 500, "CA"], ["Coke", 15000, "GA"],
["Pepsi", 3, "GA"], ["Dr. Pepper", 1000, "WY"], ["Coke", 100,
"WI"]]
>>> results_1 = election_day(election_1)
>>> print(results_1)
Coke, 15100, GA

>>> election_2 = [["cats", 150, "HI"], ["dogs", 30, "NJ"],
["dogs", 80, "AZ"], ["fish", 300, "FL"], ["cats", 50, "PA"],
["dogs", 190, "HI"], ["cats", 25, "WA"], ["cats", 75, "RI"]]
>>> results_2 = election_day(election_2)
>>> print(results_2)
cats, 300, HI
```

HW4_test.py

We have provided you with a python file called `HW4_test.py` . In this file we have created a series of tests for your usage. We understand you probably have never been exposed to testing code so you are not expected to do anything to this file or even use this file if you don't want to. However, we encourage you to use it as it is in your best interest to test your code. Feel free to add your own tests to the file to cover any additional cases you would like to test.

If you do desire to test your code, all you have to do is have the `HW4.py` and the `HW4_test.py` files in the same directory. Open and run `HW4_test.py`. After running the test, you should see the results. Check the results and start debugging if needed. If you pass all the tests you should see something like this as your output:

```
test_common_elements_1 (__main__.TestMyClass) ... ok
test_common_elements_2 (__main__.TestMyClass) ... ok
test_common_elements_3 (__main__.TestMyClass) ... ok
test_election_day_1 (__main__.TestMyClass) ... ok
test_election_day_2 (__main__.TestMyClass) ... ok
test_election_day_3 (__main__.TestMyClass) ... ok
test_find_family_1 (__main__.TestMyClass) ... ok
test_find_family_2 (__main__.TestMyClass) ... ok
test_find_family_3 (__main__.TestMyClass) ... ok
test_flatten_list_1 (__main__.TestMyClass) ... ok
test_flatten_list_2 (__main__.TestMyClass) ... ok
test_flatten_list_3 (__main__.TestMyClass) ... ok
test_flatten_list_4 (__main__.TestMyClass) ... ok
test_flatten_list_5 (__main__.TestMyClass) ... ok
test_make_odd_even_1 (__main__.TestMyClass) ... ok
test_make_odd_even_2 (__main__.TestMyClass) ... ok
test_make_odd_even_3 (__main__.TestMyClass) ... ok
test_make_odd_even_4 (__main__.TestMyClass) ... ok
test_make_odd_even_5 (__main__.TestMyClass) ... ok
test_process_string_1 (__main__.TestMyClass) ... ok
test_process_string_2 (__main__.TestMyClass) ... ok
test_process_string_3 (__main__.TestMyClass) ... ok
test_remove_duplicates_1 (__main__.TestMyClass) ... ok
test_remove_duplicates_2 (__main__.TestMyClass) ... ok
test_remove_duplicates_3 (__main__.TestMyClass) ... ok
test_unique_lists_1 (__main__.TestMyClass) ... ok
test_unique_lists_2 (__main__.TestMyClass) ... ok
test_unique_lists_3 (__main__.TestMyClass) ... ok
test_unique_lists_4 (__main__.TestMyClass) ... ok
test_unique_lists_5 (__main__.TestMyClass) ... ok

----------------------------------------------------------------------
Ran 30 tests in 0.004s
```

Read more about unittest here: [https://docs.python.org/3/library/unittest.html]

Disclaimer: The tests found in HW4_test.py are not intended to be an exhaustive list of all test cases and does not guarantee any type of grade. Write your own tests if you wish to ensure you cover all edge cases.

Authors: Soni Aggarwal and Christine Feng | Report Issues: [soni@gatech.edu or christinefeng@gatech.edu]

Grading Rubric

- `remove_duplicates`:	10 points
- `common_elements`:	10 points
- `flatten_list`:	10 points
- `make_odd_even`:	10 points
- `unique_lists`:	10 points
- `process_string:`	10 points
- `find_family`:	15 points
- `find_family`:	10 points
- `election_day`:	15 points
- **Total**	100/100 points

Provided

The following file(s) have been provided to you. There might be several, but you will only edit one of them:

1. `HW4.py`
   This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.
2. `HW4_test.py`
   This is a file containing tests for you to use if you will like to debug and test your code. You are not required to edit, submit, or even use this file.

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder and run them.

1. HW4.py

   If this file does not run (if it encounters an error while trying to run), you will get no credit on the assignment. If the file you submit is not named **exactly** like this one you will get no credit on the assignment.