Q1.) Basic Programming (5 marks)

(This will test your general Python and programming knowledge for creating a function and utilisation of some techniques of strings, lists, etc.)

Write a function `all_rounder(args)` that performs passed sequence method of a given sequence. The inputs will be in form, e.g. `all_rounder([1, 5], 'append', [3, 4, 2, 104])`. This function should create a list variable to store `[1, 5]`, then `.append()` the second list of `[3, 4, 2, 104]` to it. The final list should be returned and will be in a form of `[1, 5, [3, 4, 2, 104]]`.

The input argument storing variable `args` could include any kind of sequence types of `str`, `list`, `tuple` and `dict`. Depending on the passed sequence type, `args` should also include a method, and depending on the method some more entries.

The function `all_rounder()` should only perform one operation at a time. Test cases will never pass more than one sequence type and more than one sequence method.

Your function should return a message when a non-existing sequence method entered, and must not return an error.

Hint 1: Using `eval(), compile(), exec()` might be useful.

Hint 2: Avoid using `try … catch` and some extra libraries. This question solely be created via basic Python commands.

An example test case is given below:

```
>>> all_rounder('#', 'join', '(\'Jack\', \'Mahmut\')')
'Jack#Mahmut'
```

Q2.) NumPy (5 Marks)

(This will test the general `numpy` knowledge with a math related question.)

Write a function `padded_broadcasting(func, a, b, pad)` that takes the arguments `func, a, b` and `pad`. Here `a` and `b` are the `numpy` arrays and `func` represents the arithmetic function (e.g. `np.add`). `pad` represents the number used for padding.

For instance, if `pad=2` then the smaller array gets padded with `2`'s.

The function returns a `numpy ndarray` which is the result of the operation `func` applied to the two arrays using appropriate padding. The solution needs to work with all basic mathematical operations, that is addition, subtraction, multiplication, division, and taking a power needs to be implemented.

`a` and `b` can have a different number of dimensions.

`a` and `b` can have mismatches on one, multiple, or even all dimensions, e.g. `(6, 2, 3)` and `(5, 3, 2)`.

You may have to pad `a`, or `b`, or both arrays. Do not apply any padding if it is not necessary i.e. when standard broadcasting works.

The default argument for `pad` should be 1.

An example test case is given below:

```
>>> a = np.asarray([[1,2],[-9, 4]])
>>> b = np.asarray([[3,4,5,6],[1,1,0,-5]])
>>> padded_broadcasting(np.add, a, b, -100)
```

```
array([[   4.,    6.,  -95.,  -94.],
       [  -8.,    5., -100., -105.]])
```

Q3.) NumPy 2 - txt file analysis (5 Marks)

(For this question, a .txt file will be given. You are going to use `numpy` methods to load, read and process the file for calculations, e.g. counting the number of a given word.)

Write a function `txtanalyser(fname, t, f, sel)` that takes a `.txt` file name (`fname`) of football commentary as its input and performs various analyses for the target word (`t`).

The developed `txtanalyser(fname, t, f, sel)` function is expected to load the .txt file via `numpy` functions (`np.loadtxt()`).

The loaded file in fact has two columns for each line seperated by tabs: `'Minute'` and `'Commentary'` where the former refers to the minute in the game and the latter is the commentary in that minute. And example visual of your txt file is

```
90  Joel Matip relieves the pressure with a clearance
90  Emerson Royal puts in a cross...
90  Safe hands from Hugo Lloris as he comes out and claims the ball
```

Following this, the function should carry on funding each repetition of the target word `t` and perform calculation depending on the given (`f, sel`) pair:

- Function `f` can be any `numpy` function and should be in form `np.`, e.g. `f = np.max`.
- if `sel` is `'count'`, then `f` should automatically be assigned to `np.sum` whatever passed for `f` as the function argument. Then, for this selection, your function should calculate the number of times the target word `t` appears in the commentary column of the txt file.
- if `sel` is `'find'`, then your function should calculate the passed `f` function for the minutes column of the txt file. *Hint: Average minutes where 'RED-CARD' word was in the commentary –> f is `np.mean` and t is `'RED-CARD'`.*

Rules: - Using file I/O functions is **not permitted** - Using `pandas` for data handling and processing **is not permitted**.

An example test case is given below:

```
>>> txtanalyser('commentary.txt', 'RED-CARD!', np.mean, 'find')
59.8
```

Q4) NumPy & regex (10 marks)

Find commentary in alphabetical order (8 Marks)

In this question, your goal is to find all the commentaries whose words are arranged so that their first letters are in alphabetical order. For example:

- `Bernardo Silva walks` (because `b, s, w`)

but not

- `Bernardo Silva leaves` (because `b, s, l`, i.e., `l` comes before `s` in the alphabet)

You must implement a function called `find_alphabetical_order(fpath, check_ties=True)` that takes in two arguments:

- `fpath`: A `string` pointing to the `commentary.txt` file. Note that your function is only expected to handle the format in this file, i.e., `minute<tab>commentary`.
- `check_ties`: A `boolean` value. If set to `True`, your function must handle cases when the first letter of two consecutive words is *the same*. For example, `rearranging recovery`. Specifically, if `check_ties` is `True`, you must continue making pairwise checkings between letters in the word until you find a tie breaker (in the example, you would check first the `r`, which is a tie, then the `e`, which is another tie, then `a` vs `c`, since `a` is before `c`, we can determine that `rearranging recovery` is in alphabetical order. If this flag is set to `False`, your code

Notes:

- Your code must only look at words, not punctuation marks. In a sentence `this is a sentence, do I like it? I think so!`, you must clean out `,`, `?` and `!`.
- Your code must handle special characters like *accents* or the `Ñ` letter (like `árbitro`, `reducción` or `España`, these are Spanish words). This is because some comments may not be in English. The way to handle accents is to look for them and replace them with the non-accented version, or with an `N` in the case of `Ñ`. You must handle both upper and lower case occurrences of these characters.
- You may create additional functions to help break down some of the functionalities outside of `find_alphabetical_order()`.
- **IMPORTANT: You must use regular expressions in at least one** of the following steps:
  - clean the text from punctuation
  - turn Spanish characters into English characters
  - comparing letters

An example test case is given below:

```
>>> res = find_alphabetical_order('commentary.txt', check_ties=True)
>>> print(res[0])

Dimitrios Giannoulis hand-balls.
```

Using values in `res` answer the questions below (2 Marks)

- **Commentary languages (1 Mark)**: How many English comments did we find after calling `find_alphanumerical_order(PATH_TO_COMMENTARY_FILE, check_ties=True)`?
- **Name 3 footballers (1 Mark)**: Name 3 footballers that appear in comments returned by `find_alphanumerical_order(PATH_TO_COMMENTARY_FILE, check_ties=True)`.

Q5) Pandas (5 marks)

(Testing pandas usage and one-liner queries.)

Create a function `pd_query()` that takes no input arguments. This function should *locally* load `superheros.csv` file which stores power-related information about Marvel superheros.

Rules:

- The function `pd_query()` should implement 4 queries below and **return a data frame for each** in a `tuple`.
- All pandas query should be a **ONE-LINER**. Implementations more than 1-line will not be evaluated.

Queries:

For each, return a dataframe which stores

- (1 mark) superheros who are *good*, has greater than 80 intelligence but lower than 20 speed.
- (1 mark) average values of each column for superheros with durability value of 120.
- (1 mark) average Total values of good and bad superheros.
- (2 marks) names of superheros who has at least 3 column values better than Iron Man.

An example test case for calling the function:

```
>>> a,b,c,d = pd_query()
```