# 15-110 Fall 2019
# Midterm Exam 01

Out: Sunday 29th September, 2019 at 15:00 AST
**Due:** Sunday 29th September, 2019 at 16:20 AST

## Introduction

This midterm exam includes all the topics studied so far in the course.

**The total number of points available from the questions is 110, where 10 points are *bonus* points (i.e., you only need 100 points to get the maximum grade).**

In the handout, the file `exam01.py` is provided. It contains the functions already defined but with an empty body (or a partially filled body). You have to complete the body of each function with the code required to answer to the questions.

You need to submit to Autolab the `exam01.py` file with your code.

Only the provided *reference card* (possibly with your annotations) is admitted as a support during the exam.

The code must be written and tested using Spyder on the computers in the classroom.

# 1 Write the function code

**Problem 1.1**: (45 points)

Implement the function `clean_and_join(L, trash)` that takes as inputs a *list of strings* `L` and a *string*, `trash`. The function returns a *string* constructed as follows.

- The first two occurrences of the string `trash` in `L` are *removed* (watch out: `trash` may occur less than twice in `L`, such that you must remove up to two occurrences).
- After removing `trash`, the string `'middle␣point'` needs to be *inserted* in the midlle of the list `L`.
- All string elements from `L` are *concatenated* into a single string separated by the string `',␣'` (comma + space).
- Leading and trailing *spaces* are removed from the string.

For instance, `clean_and_join(['␣ab', 'xx', 'pq', 'rs␣␣'], 'xx')` returns the string `'ab,␣middle␣point,␣pq,␣rs'`.

`clean_and_join(['␣ab', 'xx', 'pq', 'rs␣'], 'yy')` returns the string `'ab,␣xx,␣middle␣point,␣pq,␣rs'`.

`clean_and_join([], 'yy')` returns the string `'middle␣point'`.

**Expected number of code lines:** $\leq 10$

**Problem 1.2**: (40 points)

Implement the function `list_sort(L, s)` that takes as inputs a *list of numbers*, `L`, and a string, `s`.

The function returns a tuple of *six items:*

1. the first item contains the elements of the original list `L` but in *reverse order* (e.g., if `L` is `[1, 3, 7, 2, 4, 6]`, the first item is `[6, 4, 2, 7, 3, 1]`);
2. the second item is the *maximum* value of `L` (e.g., if `L` is `[1, 3, 7, 2, 4, 6]`, the second item is `7`);
3. the third item is the *minimum* value of `L` (e.g., if `L` is `[1, 3, 7, 2, 4, 6]`, the third item is `1`);
4. the fourth item is a list is with the same elements of `L` but *sorted* according to `s`. Sorting is performed in *ascending order* if the input string `s` is `'ascending'`, and in *descending order* if `s` is `'descending'` (e.g., if `L` is `[1, 3, 7, 2, 4, 6]`, and `s` is `'ascending'`, the fourth item is `[1, 2, 3, 4, 6, 7]`);

   If `s` is neither a string nor equal to `'ascending'` or `'descending'`, sorting is done in descending order.
5. the fifth element is a list with the same elements of the `L` but taken *every three positions* (e.g., if `L` is `[1, 3, 7, 2, 4, 6]`, the fifth item is `[1, 2]`);

6. the sixth element is a list with the elements of `L` from *position index 1 to the middle point position* (e.g., if `L` is `[1, 3, 7, 2, 4, 6]`, the sixth item is `[3, 7]`).

The function also prints out the multiline string:

```
'The order was ascending'
'The difference between max and min values  is: 6'
```

where `ascending` and 6 applies to the specific used example.

Overall, for the example case `list_sort([1, 3, 7, 2, 4, 6], 'ascending')`, the returned tuple is the following: `( [6, 4, 2, 7, 3, 1], 7, 1, [1, 2, 3, 4, 6, 7], [1, 2], [3, 7] )`.

**Expected number of code lines:** $<= 15$

# 2 Complete the function code

**Problem 2.1**: (25 points)

Implement the function `nested_lists(l, i, j, k)` that takes as input a list of lists, `l`, and three integers. The function returns the `k`-th character of the `j`-th element of the `i`-th list inside `l`, if such character exists and the `j`-th element is a string. Otherwise, the function returns `None`.

For instance, if the function is invoked as `nested_lists([ [1,2,3], [2,4], ['a', 'bcd', 'e'] ], 2, 1, 1)` the returned value is `'c'`. If `j`=0, `k`=0, `i`=2, the returned value is `'a'`. For `i` = 0 (and whatever values for `j` and `k`), the returned value is `None`.

The code of the function is partially written below. You need to add the missing parts.

```
def nested_lists(l, i, j, k):
    if len(l) >=
        if len(l[i])
            if type(l[i][j]) =
                if len(l[i][j]
                    return
    return
```