Complete the following 4 problems using the following lists:

```
list1 = [5, 8, 10]
list2 = [3, 2, 9, 12, 4]
```

Problem 1. Basic operations

1. Find out the length of each list using the `len()` function.
2. Concatenate `list1` and `list2` using `+` (returns a new, concatenated, list as value). Then, find out the length of the resulting list.
3. Find the largest number in both lists. You can do it by finding the largest number in `list1`, then finding the largest number in `list2` and finally the largest of both numbers. Or you can find the largest number of the concatenated list. Try both approaches.

Problem 2. Accessing list items

1. Write an expression to add the first element of each list (must add 5 and 3 to produce 8).
2. Write an expression to add the last element of each list (10 + 4).
3. Write a sentence to replace number 8 in `list1` with 0. Check that the element has changed.

Problem 3. List Assignment

1. Run the sentence `list3 = list2`. Then, print `list3` to check its content (must be 3,2,9,12 and 4).
2. Modify `list3` so that its first element is 7. Print its content.
3. Now, print `list2`. The first has changed. This is because, due to the assignment `list3 = list2`, both variables refer to the same list.

Problem 4. Iterating lists with for loops

1. Write a loop to print each element in `list1`.
2. Write a loop to print `list2` *upside down* (starting with the last and ending with the first).

Problem 5. Sum all the elements in a list.
Write a function that receives a list and returns the sum of all the elements in the list, **computed iterating over the list with a loop**. Then, from the interpreter, create a list and invoke the function passing the list as argument. Then, print the resulting value. Compare it with what comes out of `sum(list)`.

Problem 6. Greatest Index.
Write a function that receives a list and returns the index (position) of the largest element of the list. For example, if the argument is the list `[4, 2, 7, 1, 3]` the function should return 2 since the biggest item in the list is 7 and it is in position 2 (remember that positions are numbered from zero).

Problem 7. Change Negative Numbers By Zero.
Write a function that receives a list and change all negative numbers in the list by zero. The function will return an integer indicating how many changes you have made.
For example:

```
>>> list = [ 3, -4, 5, 7, -1, 2]
>>> write_zeros(list)
2
>>> list
[3, 0, 5, 7, 0, 2]
```

Problem 8. Summation of powers
Write a function that receives a list `l` and a number `n` (integer or real, is indifferent) as parameters. This function should calculate the value of each item in the list to the power of n and add all them, returning the result. That is, if the list is composed of elements x_i, the result is calculated as $\sum {x_i}^n$.

```
sum_powers([1, 2, 3],1)  #returns 6
sum_powers([5, 11, 9], 2)  #returns 227
sum_powers([7, 0, 0], 3)  #returns 343
```

Problem 9. Given a list of ints any length, return a list with the elements "rotated left".

```
rotate_left([1, 2, 3])  #returns [2, 3, 1]
rotate_left([5, 11, 9])  #returns [11, 9, 5]
rotate_left([7, 0, 0])  #returns [0, 0, 7]
```

Problem 10. Given a list of ints, return `True` if 6 appears as either the first or last element in the list. The list will be length 1 or more.

```
first_last6([1, 2, 6])  #returns True
first_last6([6, 1, 2, 3])  #returns True
first_last6([13, 6, 1, 2, 3])  #returns False
```

Problem 11. Given a list of ints any length, return a new list with the elements in reverse order.

```
reverse([1, 2, 3])  #returns [3, 2, 1]
reverse([5, 11, 9])  #returns [9, 11, 5]
reverse([7, 0, 0])  #returns [0, 0, 7]
```

Problem 12. Return the number of even ints in the given list.

```
count_evens([2, 1, 2, 3, 4])  #returns 3
count_evens([2, 2, 0])  #returns 3
count_evens([1, 3, 5])  #returns 0
```

Problem 13. Given a list length 1 or more of ints, return the difference between the largest and smallest values in the list.
Note: the built-in `min(v1, v2)` and `max(v1, v2)` functions return the smaller or larger of two values.

```
big_diff([10, 3, 5, 6])  #returns 7
big_diff([7, 2, 10, 9])  #returns 8
big_diff([2, 10, 7, 2])  #returns 8
```

Problem 14. Return the "centered" average of a list of ints, which we'll say is the mean average of the values, except ignoring the largest and smallest values in the list. If there are multiple copies of the smallest value, ignore just one copy, and likewise for the largest value. Use int division to produce the final average. You may assume that the list is length 3 or more.

```
centered_average([1, 2, 3, 4, 100])  #returns 3
centered_average([1, 1, 5, 5, 10, 8, 7]) #returns 5
centered_average([-10, -4, -2, -4, -2, 0]) #returns -3
```

Problem 15. Consider a list in the way `[string, number, string, number, ...]` where the string are names and the number is the number of children with that name in one year. Write a function that receives such a list and returns the most popular name.

```
popular_name(['ana', 223, 'laura', 204, 'elena', 175]) #returns ana
popular_name(['laura', 204, 'ana', 223, 'elena', 175]) #returns ana
popular_name(['elena', 175, 'laura', 204, 'ana', 223]) #returns ana
```

Problem 16. Write a program to generate a vector of size n with random numbers using loops. The value of n is given by the user.

Problem 17. Write a program that, given a vector of real numbers, prints the values in `m` columns. So, if we have the vector `[1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1]` y and `m` is set to 5, then it will print:

```
1 2 3 4 5
6 7 8 9 8
7 6 5 4 3
2 1
```

Problem 18. We have a list with the votes obtained by each partie in the recent elections. Each party has assigned a positive correlative integer numbers, while 0 means a blank vote and -1 stands for an invalid vote. Let us assume we have 5 parties and a vector containing the votes (at least 15 votes), manually created by the student. Write a program to count the votes for each party, the invalid and the black votes (and also the percentages of vote), printing the results to the screen. You must use loops.

Problem 19. Write a program that generates a vector of size 15 with random numbers in range [0.0, 1.0]. The program computes the mean and the standard deviation. Each value in the vector out of the interval $[\mu - \sigma, \mu + \sigma]$ should be assigned a 0. Finally, the original vector and the final vector must be printed in 5 columns format.

Problem 20. Storing a matrix as a vector in a list. So imagine we have a matrix of size $ROWS \times COLUMNS$. We can store that as a vector in a list provided that:

- Your vector is of size $SIZE = ROWS \times COLUMNS$.
- You select an arrangement. For instance, let's select to arrange by row vectors. If we have a 2×3 matrix, we will have a vector such as $[a_{1,1}, a_{1,2}, a_{1,3}, a_{2,1}, a_{2,2}, a_{2,3}]$, with $a_{i,j}$ being the element at row i column j in the matrix.
- Then, each element $a_{i,j}$ at row i column j in the matrix is indexed in the vector with position $p = i \times COLUMNS + j$.

So, now, here are the tasks:

1. Write a function that creates and returns a matrix of size 3×4 and initialize each element $a_{i,j} = i * j$.
2. Write a function that creates and returns a matrix of size 3×4 and initialize each element $a_{i,j} = i + j$.
3. Write a function that prints a matrix to the screen preserving the matrix form. For instance, a matrix of size $2 * 3$ would be printed as:

```
1 2 3 4
5 6 7 8
```

4. Write a function that receives two matrices (of the same size) and returns the sum of the two of them (matrix addition).
5. Write a function that receives two matrices (the former of size $n \times m$ and the second of size $m \times p$. The function shall return the matrix product.

Create a program that uses all the above defined functions.

Problem 21. Write a program to, firstly, compute and store the multiplication tables for 1 to 10, that is, 100 values! Then, the program must print the multiplication table to the screen.

Problem 22. Function lastChars, receiving two strings and returning a string with the first symbol of the former and the last one from the latter. If either string is length 0, use '@' for its missing char.

```
lastChars("last", "chars") #returns "ls"
lastChars("yo", "java") #returns "ya"
lastChars("hi", "")  #returns "h@"
```

Problem 23. Function makeTags the web is built with HTML strings like `"<i>Yay</i>"` which draws `"Yay"` as italic text.

```
makeTags("i", "Yay") #returns "<i>Yay</i>"
makeTags("i", "Hello") #returns "<i>Hello</i>"
makeTags("cite", "Yay") #returns "<cite>Yay</cite>"
```

Problem 24. Function `rotateleft(string, pos)` given a string, return a "rotated left x" version where the first x chars are moved to the end. The string length will be at least 2. or an error will raise.

```
rotateleft("Hello", 2) #returns "lloHe"
rotateleftleft("java", 3) #returns  "ajav"
rotateleft("Hi",2) @returns "Hi"
```

Problem 25. Write a function that implements a substitution cipher. In a substitution cipher one letter is substituted for another to garble the message. For example A could be substituted by Q, B by T, C by G etc.

Your function should take two parameters, the message you want to encrypt, and a string that represents the mapping of the 26 letters in the alphabet. Your function should return a string that is the encrypted version of the message. For this exercise, limit the strings to include only alphabetical symbols and white spaces.

```
substitution("abc","cdbafgehmnlijtsrqpozyxwv") #returns "cdb"
```

Problem 26. Write a function that decrypts the message from the previous exercise. It should also take two parameters. The encrypted message, and the mixed up alphabet. The function should return a string that is the same as the original unscripted message. For this exercise, limit the strings to include only alphabetical symbols and white spaces.

Problem 27. Write a function called `rot13` that uses the Caesar cipher to encrypt a message. The Caesar cipher works like a substitution cipher but each character is replaced by the character 13 characters to 'its right' in the alphabet. So for example the letter 'a' becomes the letter 'n', 'b' becomes 'o', etc. If a letter is past the middle of the alphabet then the counting wraps around to the letter 'a' again, so 'n' becomes 'a', 'o' becomes 'b' and so on.
Note: Whenever you talk about things wrapping around its a good idea to think of the arithmetic remainder operation from the integer division.

Problem 28. Using the methods `count` and `lower`, and the operator `in`, write a function called `count_vowels` for counting the occurrences of each vowel in a given text. The function should return a list with the vowel's occurrences, from 'a' to 'u'.

So, `"Bugs Bunny! Bugs Bunny! RAH RAH RAH!"` contains 3 a's, 0 e's, 0 i's, 0 o's and 4 u's; thus the function shall return `[3, 0, 0, 0, 4]`.

Problem 29. Now, let's do the same but for all the letters from 'a' to 'z'. Therefore, write a function called `count_letters` that receives a string and returns a list with the occurrences of each letter from 'a' to 'z'.
Hint 1: forget about ñ.
Hint 2: remember that the built-in function `ord` returns the ASCII code of a symbol, and

the built-in function `chr` returns the symbol for a given ASCII code. That is, `ord('-')` return `95` and `chr(95)` returns `'-'`.

Problem 30. Write the function `metamorphosis` for automatically converting from a string to any valid data type among the following:

- integer: just containing digits plus leading and ending spaces,
- bool: True or False plus leading and ending spaces,
- float: digits+dot+digits format, allowing ' 23.4 ', ' .234 ' and ' 234.' as valid floats,
- str: the remaining cases it is just a string!

Therefore, the following lines give you some example of how to call the function with valid values, as well as the requested outputs:

```
metamorphosis(' 23 ') # returns the integer 23
metamorphosis(' 2 3 ') #is not an integer, it will return a str
metamorphosis(' 23. ') #returns the float number 23.0
metamorphosis(' 23 .') #is not a number, it will return a str
metamorphosis(' True ') #returns the boolean True
metamorphosis(' T r ue') #returns a str.
```

Problem 31. Using this variables

```
names = "Deep Toot, Wikipedia Brown, Toots Magoots, George Washingturd, Zeus Toots, Queen
↪  Kong, John Marmalade, Mr. E. Sir, Smalls Assquatch, and Joe Bag O' Doughnuts"
jobs = "Dog Food Tester, Note Taker for College Students, Gum Buster, Phone Psychic,
↪  Chicken Sexer, Jelly Doughnut Filler, Stand-in Bridesmaid, Golf Ball Diver, Odor
↪  Judges, Potato Chip Inspector, and Egg Inspector"
places = "Okay (Oklahoma, USA), Tubbercurry (Ireland), Boring (Oregon, USA), Dull
↪  (Scotland), Fuckersberg (Austria), Scratchy Bottom (Devon, UK), Tyewhoppety
↪  (Kentucky, USA), Ken-Taco-Huts (New Mexico, USA), Ding Dong (Texas, USA),
↪  Blubberhouses (Yorkshire, UK), Satan's Kingdom (Vermont, USA), Moron (Mongolia,
↪  Russia), The Finger Lakes (NY, USA), and Coxsackie (NY, USA)"
fees = [300, 35, 45, 64, 120, 95, 65, 80, 250, 76, 98]
```

write the following functions:

- `str_to_list` receiving a text and returning a list of strings according to the following rules: i) eliminate any occurrence of the text "and", ii) separate the strings using the comma symbol (','), iii) eliminate any leading or ending white space.

  If the function is called with the names variable, the returning list must be: `["Deep Toot", "Wikipedia Brown", "Toots Magoots", "George Washingturd", "Zeus Toots", "Queen Kong", "John Marmalade", "Mr. E. Sir", "Smalls Assquatch", "Joe Bag O' Doughnuts"]`.

- `city_and_country` that receives the string corresponding to one of the valid places and returns a list with two strings: the city name and the country at positions 0 and 1, correspondingly. For instance, if the function is called with `"The Finger Lakes (NY, USA)`, it should return `['The Finger Lakes','USA']`.

- `fusion` receiving four objects: a name, a job, a place and a fee. The function should return a list with the four values in the given order. For instance, if this function is called `fusion('Deep Toot', 'Egg Inspector', [' The Finger Lakes', 'USA'], 350)`, then it should return `['Deep Toot', 'Egg Inspector', ' The Finger Lakes -USA-', '350']`.

- `merge` this function receives 4 lists containing names, jobs, places and fees, respectively. This function should generate a 100 elements list, each element contain a list with a random name, a random job, a random place and a random fee. As an example, if the function receives the four lists names, jobs, places and fees, one element of the list to return might be: `['Deep Toot', 'Egg Inspector', ' The Finger Lakes -USA-', '350']`.

- `list_to_str` receiving a list L of elements like that generated in function merge and returns a string that i) converts the list L to a list of strings by joining the each of its elements with the comma character (',') -maybe it is interesting to remember you the str's method join; and ii) joining all the elements of L with the character new line (`'\n'`). Therefore, if the first two elements of L are `[['DeepToot', 'EggInspector', ' TheFingerLakes-USA-', '350'], ['John Marmalade', 'Gum Buster', ' Tyewhoppety -USA-', '45'], ...]`, then the returning string should start with `'Deep Toot,Egg Inspector,The Finger Lakes -USA-,350\nJohn Marmalade,Gum Buster,Tyewhoppety -USA-45\n...'`

- `count_letters` that should return the number of letters within the name (do not including the surname letters!).

Write a program that calls the required functions in order to print to the screen 100 lines, each line indicating who was doing a certain job, where it was done and the corresponding fees.