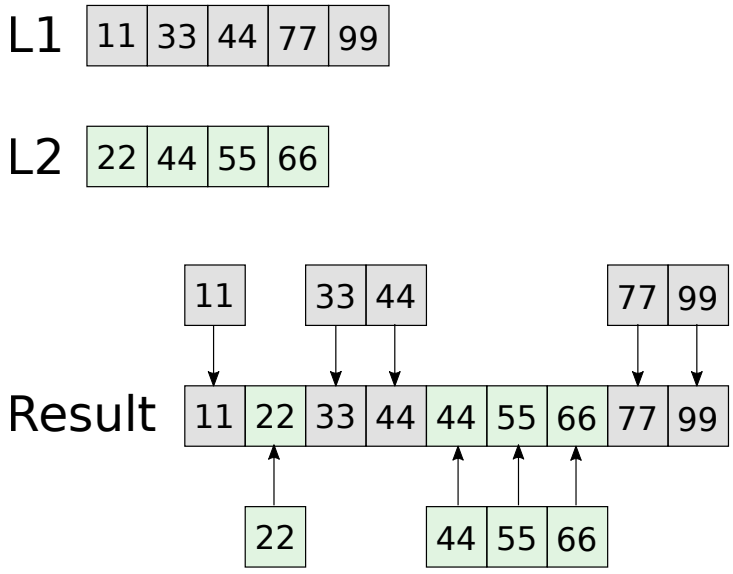
Exercise 8

Programs should be written in the programming language Python according to the respective exercise description. The program must be correct in terms of syntax and semantic. If there exists a minimal solution calling only a single pre-defined function, this function is not allowed.

The weekly exercise should be uploaded on ILIAS as a single Jupyter Notebook (.ipynb). Processing time of each exercise runs from Wednesday, 8:00 until following Tuesday, 8:00, if not communicated differently. Make sure that answers to questions are contained within the Jupyter Notebook that is uploaded on ILIAS.

1. Assume, you have two random lists of integer objects $L1$ and $L2$. The elements in the two lists are pre-sorted numerically. Write a program that puts the elements from the two lists into the list *Result* in the correct order. Do not use any predefined sort-functions!

Example:



2. Write a program that determines all l-mers with length $l = 3$ (words of size 3) present in a given sequence `GACAGACCAG` and prints them as shown in the example below.

Ouptut:

```
GACAGACCAG
GAC
 ACA
  CAG
   AGA
    GAC
     ACC
      CCA
       CAG
```

3. Write a program based on task 2 that prints all l-mers with their respective frequencies. Implement a function that takes a sequence and the l as arguments and returns the a dictionary with the respective frequencies.

Ouptut:

```
Sequenz: GACAGACCAG
l = 3

l-mers frequencies:
GAC 2
ACA 1
CAG 2
AGA 1
ACC 1
CCA 1
```

4. Write a program based on task 3. The output should now also be saved to a file named `3mer_frequencies.txt`.

5. The file `cognames.txt` (ILIAS) contains for each gene group identifier a tab-separated list containing the `COG`, a one letter abbreviation of the corresponding functional category `fun` (see `fun.txt` for definitions), and the gene product `name`.

Write a program that reads the information and stores it in a dictionary (`COG` as key, tuple of `func` and `name` as value. The complete dictionary should be printed sorted by `COG` in reversed order.

Ouptut:

```
COG5665 K CCR4-NOT transcriptional regulation complex, NOT5 subunit
COG5664 O Predicted secreted Zn-dependent protease
COG5663 S Uncharacterized protein, HAD superfamily
COG5662 K Transmembrane transcriptional regulator (anti-sigma factor RsiW)
COG5661 O Predicted secreted Zn-dependent protease
...
```

6. Write a program based on task 5. Read in a `COG` from standard input. The program should print the abbreviation of the functional category and the gene product name based on the completed dictionary for the respective `COG`. Note the exact way in which the output is generated in the example below.

Example:

```
Provide a COG ID: COG3855
```

Ouptut:

```
COG3855 (G) Fructose-1,6-bisphosphatase
```

7. Write a program based on task 5. Based on the complete dictionary, we now want to print all `COG` identifiers with their respective functional category and gene product name, that are ribosomal proteins (name contains `Ribosomal protein`. Verify your output.

Ouptut:

```
COG0048 J Ribosomal protein S12
COG0049 J Ribosomal protein S7
COG0051 J Ribosomal protein S10
COG0052 J Ribosomal protein S2
...
```

8. Write a program based on task 7. Read a search text from standard input. Based on the complete dictionary, all respective entries, for which the gene product contains the search text should be printed.

Example:

```
Gene product search text: aminotransferase
```

Ouptut:

```
COG0001: (H) Glutamate-1-semialdehyde aminotransferase
COG0075: (EF) Archaeal aspartate aminotransferase or a related aminotransferase, ...
COG0079: (E) Histidinol-phosphate/aromatic aminotransferase or cobyric acid decar...
COG0115: (EH) Branched-chain amino acid aminotransferase/4-amino-4-deoxychorismat...
COG0160: (E) 4-aminobutyrate aminotransferase or related aminotransferase
COG0161: (H) Adenosylmethionine-8-amino-7-oxononanoate aminotransferase
COG0436: (E) Aspartate/methionine/tyrosine aminotransferase
COG1167: (KE) DNA-binding transcriptional regulator, MocR family, contains an ami...
COG1364: (E) N-acetylglutamate synthase (N-acetylornithine aminotransferase)
COG1448: (E) Aspartate/tyrosine/aromatic aminotransferase
COG1932: (HE) Phosphoserine aminotransferase
COG3977: (E) Alanine-alpha-ketoisovalerate (or valine-pyruvate) aminotransferase
COG4992: (E) Acetylornithine/succinyldiaminopimelate/putrescine aminotransferase
```

9. Let us assume, we have the following character string `"((1:0.2,(2:0.33,3:0.4)4:1)5:0.1)6;"` representing a phylogenetic tree in Newick format. We want to extract all the numbers present in that character string right after the colon characters. These numbers represent the branch lengths. Write a program that extracts the numbers based on regular expression(s). Subsequently calculate and print the sum of branch lengths.

Ouptut:

```
The extracted numbers are 0.2, 0.33, 0.4, 1, 0.1.
The sum of branch lengths is 2.03.
```