I. Задание для выполнения в классе:

1. Загрузить в среду архив cta_seg.zip
2. Распаковать, запустив ячейку ниже

```
!unzip cta_seg.zip
```

3. Написать функцию, которая принимает на вход имя папки (в данном случае cta_seg) и уровень seg (в данном случае B1 или Y1), ищет там все файлы .seg соответствующего уровня и возвращает их список.

```
import glob
import os


def list_segs(path, level):
    return glob.glob(os.path.join(path, f"*.seg_{level}"))


list_segs("cta_seg", "B1")
```

4. Модифицировать функцию так, чтобы она открывала каждый найденный файл .seg, читала оттуда названия меток (кроме пустых) и записывала их в текстовый файл - метки из каждого файла .seg на отдельной строке, разделённые пробелами.

```
from itertools import product
letters = "GBRY"
nums = "1234"
levels = [ch + num for num, ch in product(nums, letters)]
level_codes = [2 ** i for i in range(len(levels))]
code_to_level = {i: j for i, j in zip(level_codes, levels)}
level_to_code = {j: i for i, j in zip(level_codes, levels)}


def read_seg(filename: str, encoding: str = "utf-8-sig") -> tuple[dict, list[di
    with open(filename, encoding=encoding) as f:
        lines = [line.strip() for line in f.readlines()]

    # найдём границы секций в списке строк:
    header_start = lines.index("[PARAMETERS]") + 1
    data_start = lines.index("[LABELS]") + 1

    # прочитаем параметры
    params = {}
    for line in lines[header_start:data_start - 1]:
```

```
            key, value = line.split("=")
            params[key] = int(value)

        # прочитаем метки
        labels = []
        for line in lines[data_start:]:
            # если в строке нет запятых, значит, это не метка и метки закончились
            if line.count(",") < 2:
                break
            pos, level, name = line.split(",", maxsplit=2)
            label = {
                "position": int(pos) // params["BYTE_PER_SAMPLE"] // params["N_CHAN
                "level": code_to_level[int(level)],
                "name": name
            }
            labels.append(label)
        return params, labels


    def get_seg_contents(path, level):
        files = list_segs(path, level)
        res = []
        for file in files:
            _, labels = read_seg(file)
            res.append(" ".join(i["name"] for i in labels if i["name"]) + "\n")
        with open(path + "_" + level + ".txt", "w") as f:
            f.writelines(res)


    get_seg_contents("cta_seg", "B1")
```

Задание для выполнения в классе:

Написать функцию, которая принимает на вход два имени файлов .seg и возвращает список списков следующего формата:

```
[
    [[метка1_1, метка1_2], [метка2_1, метка2_2, ...]],
    [[метка1_2, метка1_3], [метка2_5, метка2_6, ...]],
    ...
]
```

где `метка1_*` - это метки из первого файла, а `метка2_*` - метки из второго, которые лежат **между** соответствующими метками из первого. Используйте циклы.

Считайте, что интервал, ограниченный двумя метками, полуоткрытый - т.е. если `метка2_2` совпадает с меткой `метка1_1`, она будет входить в интервал [`метка1_1`, `метка1_2`), а если совпадает с меткой `метка1_2` - не будет.

```
def match_seg_levels(filename_upper, filename_lower):
    params, labels_upper = read_seg(filename_upper, encoding="cp1251")  # жёлтые метки в код
    _, labels_lower = read_seg(filename_lower)
    res = []
    ctr = 0
    for start, end in zip(labels_upper, labels_upper[1:]):
        res.append([[start, end], []])
        for label in labels_lower[ctr:]:  # начнём не с начала, а с того места, где останови
            if start["position"] <= label["position"] < end["position"]:
                ctr += 1
                res[-1][-1].append(label)
            elif end["position"] <= label["position"]:  # если слово закончилось, дальнейшие
                break
    return res


match_seg_levels("cta_seg/cta0001.seg_Y1", "cta_seg/cta0001.seg_B1")
```

2. Модифицируйте функцию так, чтобы результат она записывала в текстовый файл в следующем формате:

```
<время метки1_1 в секундах> <время метки1_2 в секундах> <имя метки1_1> <имя метки2_1>
```

Например:

```
0.0 0.412 юрий j u0 r' i4
0.412 1.24 трифонов t r' i0 f a4 n a4 f
```

```
def match_words_to_sounds(filename_upper, filename_lower, res_filename="res.txt"):
    params, labels_upper = read_seg(filename_upper, encoding="cp1251")
    _, labels_lower = read_seg(filename_lower)
    res = []
    ctr = 0
    for start, end in zip(labels_upper, labels_upper[1:]):
        if not start["name"]:
            continue  # паузы нас не интересуют
        start_time = round(start["position"] / params["SAMPLING_FREQ"], 3)
        end_time = round(end["position"] / params["SAMPLING_FREQ"], 3)
        labels = []
        for label in labels_lower[ctr:]:
            if start["position"] <= label["position"] < end["position"]:
                ctr += 1
                labels.append(label)
```

```
            elif end["position"] <= label["position"]:  # оптимизация
                break
        label_names = [i["name"] for i in labels if i["name"]]
        res.append(f"{start_time}\t{end_time}\t{start['name']}\t" + "\t".join(label_names) +
    with open(res_filename, "w") as f:
        f.writelines(res)


match_words_to_sounds("cta_seg/cta0001.seg_Y1", "cta_seg/cta0001.seg_B1")
```