HLPL1 Final test, 11/05/2023, 14:00

Conditions (from the Moodle course)

You get 45 minutes to solve the exercises using **one instance** of Windows CMD and Visual Studio Code. Only one source code and the input.csv and output.csv files can be opened in VS Code; thus, you should close all the opened tabs before starting the test. **You should use folder exam on the desktop. It is the only folder which content can be opened during the test. Opening files from other locations will be considered as cheating. You can start the test within 3 minutes delay. The deadline for the submission is set in the Moodle system, and it cannot be extended. Non-uploaded solutions will result in a score of 0.** You can open a browser and log in to Moodle **only in the last 5 minutes**. The source code should be uploaded to the available assignment. We recommend using the *drag&drop* method to submit it. **You should sign the exercise sheet and leave it on the table. Moreover, the VS Code (including your source codes) should be kept open.** If we cannot find the signed sheet, your score will be 0. After finishing the test, you should leave the room immediately. Instructors cannot answer questions due to the strict schedule of the tests and administrative steps. You should present your ID. **Double check that you switch off all your electronic devices and place them into your bag or pockets. No devices can be worn (e.g., earplugs or smart watches) or placed on the table/desktop. Only extension *C/C++ for Visual Studio Code* can be enabled in VS Code.**
You will get one instance of the official reference page: https://arato.inf.unideb.hu/panovics.janos/stdc.pdf. We will reuse them, so you cannot write on them. We use the control system during the test, so we can check, record, unlock, or control your screens. Starting the test, you automatically declare that you have read and accepted the conditions. We strictly follow the regulations.
You will get a single exercise which will be similar to the given sample. Each exercise scores at most **36 points**, and you can get subscores. A passing solution should fulfill the following requirements (more details can be found in document Final test // Grading): Do not contain a syntax error; Get at least 50% of the points (18 points); Get all the mandatory points (12 points, marked with Y). Error messages should be printed to the standard error channel, and each message should contain its cause.

Declaration

I declare with my signature that I read the conditions in Moodle and understood them.

Name:	Neptun-code:	Signature:

Defining record

Define the record `TRACK` based on a structure having the following fields:

- `title`: title of the track (a string having at most `40` characters, type: `char[]`)
- `length`: length of the track (in seconds, type: `int`)
- `plays`: number of plays (in hundred-thousands, type: `int`)

Reading records from the file

Open and read the CSV file, which name is passed as the first command-line argument! Each line represents a single `TRACK` record in the following format:

```
<title>;<length>;<plays>
```

Notes:

1. Solve the exercise in function `main()`.
2. The end of the file is denoted by `EOF`.
3. It is guaranteed that the file contains at least `1` and at most `30` lines.
4. Each line contains at most `60` characters.
5. The file uses semicolons (`';'`) as the delimiter.
6. Print an error message and exit with status code `1` if the command-line argument is not present.
7. Print an error message and exit with status code `2` if the file does not exist.

Sorting records

Sort the array using built-in function `qsort()`, and the following stages:

1. number of plays (descending)
2. length of the track (ascending)
3. title of the track (ascending)

Notes:

1. Invoke function `qsort()` in function `main()`.

Writing records to file

Open and write the CSV file, which name is passed as the second command-line argument and write the sorted sequence of records to the file!

Notes:

1. Solve the exercise in function `main()`.
2. The file should have the same format as the input file.
3. Print an error message and exit with status code `3` if the command-line argument is not present.
4. Print an error message and exit with status code `4` if the file cannot be opened.

Querying the records

Define function `query`, which gets the memory address of a `TRACK` array and its length, then returns the average of the tracks' lengths.

Formal parameter list

1. `tracks`: the memory address of the array (type: `TRACK *`)
2. `length`: the length of the array (type: `int`)

Returned value

the average of the tracks' lengths (type: `double`)

Calling the function

1. Call the function in function `main()`, passing the memory address of the array and its length.
2. Print the returned value to the standard output.

Sample execution

Command-line arguments

```
input.csv output.csv
```

Content of file `input.csv`

```
Dua Lipa - Physical;193;810
David Guetta - Memories;210;682
Calvin Harris - Miracle;186;810
Miley Cyrus - Flowers;200;977
```

Standard output

```
197.250000
```

Content of file `output.csv`

```
Miley Cyrus - Flowers;200;977
Calvin Harris - Miracle;186;810
Dua Lipa - Physical;193;810
David Guetta - Memories;210;682
```