CSE 102 Programming Assignment 2

DUE

October 25, 2018, 23:55

Description

- This is an individual assignment. Please do not collaborate
- If you think that this document does not clearly describe the assignment, ask questions before its too late.
- This assignment is about simple file read/write and array usage. You can use pointers and functions. You cannot use dynamical memory allocation. You cannot use recursion. You can use global variables.

This is a `C` Programming assignment. You will write a `C` program according to the following description.

- The program reads real numbers from a file. Determines the chunks according to a criteria. For each chunk, the average of the numbers is calculated. Chunks are listed in ascending order based on their average.
- Input file contains a sequence of real numbers which are separated by whitespace. The whole sequence consists of chunks which are going to be separated by the following criteria:
  - `n` is the number to be added and `a` is the average of the chunk. If `n` is added to the current chunk the average becomes `b`.
  - `n` is going to be added to the chunk if the following condition is satisfied:
  - `!(b > a*(1+p1) || b < a*(1-p1) || a > n*p2 || a < n/p2)`
  - Beware, there is a `!` at the beginning of the condition.
  - Here, `p1` and `p2` are pre-defined variables. Declare them as constants in your program.
- Each line of the output file lists the numbers in chunks.
- Code it so that it reads a text file named `input.txt` and writes to a text file named `output.txt`. (If you don't follow this convention your grade will be `0.0`).

Example

- Contents of the input file:

```
12.432 23.5 344.6 11.85 2.5 8.2313 19.27 70.001 23.64 13.62
```

- Given `p1 = 0.5` and `p2 = 20`, there are 4 chunks:
- Find averages of numbers in each chunk.

```
12.4320 23.5000                  ave: 17.9660
344.6000                         ave: 344.6000
11.8500 2.5000 8.2313 19.2700    ave: 10.4628
70.0010 23.6400 13.6200          ave: 35.7537
```

- Create a text file with chunks ordered as stacked lines based on the calculated average of each.

```
11.8500 2.5000 8.2313 19.2700
12.4320 23.5000
70.0010 23.6400 13.6200
344.6000
```

- Each line is a chunk. Row order is according to the average(the chunk with the smallest average is on the first row.)

Remarks

- Maximum length of the input sequence is `1000`.
- Maximum number of chunks is `100`.
- Minimum length of a chunk is `1`.
- The first number in the sequence belongs to the first chunk.
- There is at least one chunk in the sequence.
- Sequence starts with a chunk.
- You don't have to do error checking on the input file. You can safely assume that you will be given a proper input file which doesn't violate the described format.
- Be careful with the size of the array you allocate in program stack. Large arrays may not fit in program stack(stack size may be smaller on the test machine) and your program crashes.
- Make sure you can read input files with or without a tailing newline at the end. (If you are using a windows machine, newline is `CRLF`, on unix it is `LF`). You can alter this using advanced editors (i.e. Visual Studio Code). Test your code for every possible combination.
- Do not print anything other than the expected output. (For this assignment, your program prints NOTHING).
- You cannot use anything which is not covered in class.
- Do not submit any of the files you used for testing.
- Do not submit your output file.

Turn in:

- Source code of a complete C program. Name of the file should be in this format: `<full_name>_<id>.c`.
- Example: `gokhan_kaya_000000.c`. Please do not use any Turkish special characters.
- You don't need to use an IDE for this assignment. Your code will be compiled and run in a command window.
- Your code will be compiled and tested on a Linux machine(Ubuntu). GCC will be used.
- Make sure you don't get compile errors when you issue this command : `gcc <full_name>_<id>.c`.
- A script will be used in order to check the correctness of your results. So, be careful not to violate the expected output format.
- Provide comments unless you are not interested in partial credit. (If I cannot easily understand your design, you may loose points.)
- You may not get full credit if your implementation contradicts with the statements in this document.

Late Submission

- Late submission is NOT accepted.

Grading (Tentative)

- `Max Grade` : `100`.

All of the followings are possible deductions from `Max Grade`.

- No submission: `-100`. (be consistent in doing this and your overall grade will converge to `N/A`) (To be specific: if you miss `3` assignments you'll get `N/A`)
- Compile errors: `-100`.
- Irrelevant code: `-100`.
- Major parts are missing: `-100`.
- Unnecessarily long code: `-30`.
- Using language elements and libraries which are not allowed: `-100`.
- Not caring about the structure and efficiency: `-30`. (avoid using hard-coded values).
- Significant number of compiler warnings: `-10`.
- Not commented enough: `-5`. (Comments are in English)
- Source code encoding is not `UTF-8` and characters are not properly displayed: `-5`. (You can use 'Visual Studio Code', 'Sublime Text', 'Atom' etc... Check the character encoding of your text editor and set it to `UTF-8`)

- Fails at properly reading the file: `-30`.
- Fails during file write: `-30`.
- The order is wrong: `-30`.
- Averages and chunks are wrong: `-30`.
- Infinite loop: `-90`.
- Prints anything other than the expected: `-30`.
- Unwanted chars and spaces in `output.txt`: `-30`.
- Submission includes files other than the expected: `-10`.
- Submission does not follow the file naming convention: `-10`.
- Sharing or inheriting code: `-200`.

Note: Some of these items are not independent. So, you cannot expect isolation of many of them. For example, if you cannot read input file correctly, you will fail to produce the correct output file. Partial grading is not guaranteed.