CSE 102 Programming Assignment 2

DUE

November 4, 2022, 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.

You won't be given a chance to correct any mistakes.

- Write a C code which does the following:
  - Read `input.txt`
  - `input.txt` may store `100` **different** numbers at most. But there is no limitation about the number of integers repeated. You will read till the end of file.
  - Count the repetitions of the integers
  - Write the repetitions to `output.txt` in ascending order. (correct ordering is `30pts`)

Example

- `input.txt`:

```
12 10 12 12    5   10 5 5 5   5
-8   5          5      10      4  -8 4
```

- Your program counts the repetitions of integers and writes the counted values to `output.txt`
- `output.txt` after the execution:

```
-8:2
4:2
10:3
12:3
5:7
```

Another Example

- `input.txt`:

```
5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

- Your program counts the repetitions of integers and writes the counted values to `output.txt`
- `output.txt` after the execution:

```
5:110
```

Ordering (30pts)

`output.txt` should have the numbers in ascending order(with respect to their counts). The number which repeats the least should appear first. If the counts are the same, the smallest number should appear first.

Remarks

- You can only create arrays of size `100` or less. If you create bigger arrays you get `0.0`. Code efficiently. You may lose points if you create unnecessary arrays.
- Be careful with the name of the files (`input.txt` and `output.txt`). You won't be given a second chance if you make a mistake about this. Your program will fail and your grade will be `0.0`.
- Do not use any elements which is not covered in class.
- Do not submit your code without testing it with several different scenarios.
- Write comments in your code.
- You can use `ftell()`, `fseek()` and other useful functions for file read/write operations.
- There can be negative and positive integers.
- You can assume that the file is error-free. (i.e. there are only negative and positive integers in the file.)
- Properly check end-of-file and successful read/write operations.
- Efficiency of your implementation is important. Comment about the efficiency of your code. (`10 pts`)

Turn in:

- Source code of a complete C program. Name of the file should be in this format: `<full_name>_PA2.c`.
- Example: `james_clerk_maxwell_PA2.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 that your program does not require specific encodings/markings/line-ending-chars. Make sure it works with a file created in a linux environment.
- Make sure you don't get compile errors when you issue this command : `gcc <full_name>_PA2.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

- Not accepted.

Grading (Tentative)

- `Max Grade` : `100`.
- Multiple tests(at least `5`) will be performed.

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

- use macros instead of hard-coded values, otherwise you may lose: `-10`.
- No submission: `-100`.
- Compile errors: `-100`.
- Irrelevant code: `-100`.
- Major parts are missing: `-100`.
- Unnecessarily long code: `-30`.
- inefficient implementation: `-20`.
- Using language elements and libraries which are not allowed: `-100`.
- Not caring about the structure and efficiency: `-30`. (avoid using hard-coded values, avoid hard-to-follow expressions, avoid code repetition, avoid unnecessary loops).
- 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`).
- Missing or wrong output values: `Fails the test`.
- Output format is wrong: `-30`.
- Infinite loop: `Fails the test`.

- Segmentation fault: `Fails the test`.
- Fails 5 or more random tests: `-100`.
- Fails the test: `deduction up to 20`.
- Prints anything extra: `-30`.
- Requires space/newline at the end of the file: `-20`.
- Requires specific newline marking (CR/LF): `-20`.
- Unwanted chars and spaces in output: `-30`.
- Submission includes files other than the expected: `-10`.
- Submission does not follow the file naming convention: `-10`.
- Sharing or inheriting code: `-200`.