CSE 102 Programming Assignment 5

DUE

December 15, 2021, 23:55

Description

- The simplest assignment of the semester.
- 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:
- Your program will read a file named `input.txt`
- `input.txt` contains single line text. The text describes a tree structure. It follows the `Newick` format rules. But there are simplifications and/or modifications:
  - Each node is represented by a single character. They are separated by `comma`. Branching is described by matching parentheses.
  - The length of the text will not be greater than `250`
- Below is an example of a tree description:

  ```
  (A,(c,B,e),K,D,e,(f,(d,F)))
  ```

- This tree can be visualized as follows:

  ```
  -A
  --c
  --B
  --e
  -K
  -D
  -e
  --f
  ---d
  ---F
  ```

- Another example:

  ```
  (A,(A,A,(A),A),A)
  ```

- Visualization:

  ```
  -A
  --A
  --A
  ---A
  --A
  -A
  ```

- Another example:

```
A,(A,A,(A),A),A
```

- Visualization:

```
A
-A
-A
--A
-A
A
```

- Your program will read the simplified newick formatted tree description from a file and write the visualization of the described tree to `output.txt`.
- Core part of your implementation (parsing and creating visualization) should utilize **recursion** otherwise you will get `0pts`. For this problem, coding without recursion is significantly easier.
- Pay attention to the structure of the output. If your program prints something slightly different or anything extra, you will loose at least `30pts`. Don't print any debug messages or greeting texts. Don't add extra spaces, newlines, commas etc...

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 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>_<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.
- If your program requires additional compile and link options, state that requirement at beginning of your source code as a comment.

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`.

- `#define HARD_CODED_VALUES -10`.
- 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`.

- 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`.