CSE 102 Programming Assignment 4

DUE

November 26, 2024, 23:55

Description

- This is an individual assignment. Please do not collaborate
- If you think that this document does not clearly describes the assignment, ask questions before its too late.

For this assignment, you are expected to write an interactive program which, according to the user input, creates dynamic responses and do simple calculations.

You have to use recursion in the main parts of the solution.

`Module`

- A module has an identifier. (Unique. Max 30 chars. Does not start with a digit.)
- A module has an arithmetic description and returns a scalar value based on the given description.
- `<scalar>*<module_id>+<scalar>*<module_id>+...`
- Every term in the description starts with a scalar value. This value is a positive integer.
- In between the terms there is a `+` sign.
- If a scalar is followed by a `*` sign, there is a module_id in the term.
- If there isn't a `*` in the term, the term is just a scalar.
- There can be up to `100` terms in the description.

Example:

- Lets start with a module which has the following identifier
- `module1`
- Assume that this module has the following description:
- `5*blabla+3+2*module0+1*another_module+55`
- With the description above, we are introduced with 3 other modules: `blabla`, `module0` and `another_module`.
- In addition to these modules, there are scalar values in the description.
- The return values of modules will be multiplied by scalars and the result will be added to a sum and this will be the result of `module1`. (Scalar values will also be added to the sum, of course)
- Your program calculates and prints the return value of the first module given by the user.

A Full Example

```
> Module name?:                                  >> Your program prints
> module1                                        >> User types
> Define module1:                                >> Your program prints
> 5*blabla+3+2*module0+1*another_module+55       >> User types
> Define blabla:                                 >> Your program prints
> 3*more_blabla+400                              >> User types
> Define more_blabla:                            >> Your program prints
> 12+11+2                                        >> User types
> Define module0:                                >> Your program prints
> 100                                            >> User types
> Define another_module:                         >> Your program prints
> 100*last_module+5*new_module                   >> User types
> Define last_module:                            >> Your program prints
> 50                                             >> User types
> Define new_module                              >> Your program prints
> 10                                             >> User types
> 7683                                           >> Your program prints
```

The last line is the result of `module1`

Another Example:

```
> Module name?:
> A
> Define A:
> 1*B+1*C+1*D
> Define B:
> 1*E
> Define E:
> 1
> Define C:
> 1
> Define D:
> 1*F
> Define F:
> 1
> 3
```

The last line is the result of `A`

Remarks

- Your program calculates and prints the return value of the first module given by the user.
- You have to use recursion in order to calculate the return value of a module.
- You will calculate the result on-the-fly. You don't need a tree. You don't know the depth of the tree or the nodes until you ask all the questions to the user.
- There may be spaces in between the terms of the user input. Below are valid inputs:

```
67+      10+5*  module_x      +2   *   module_y
```

- **You are not allowed to use global variables.**
- Do not submit your code without testing it with several different scenarios.
- You can use `struct` and/or `union`
- You **cannot** use dynamic allocation. (You have to use partially-filled arrays)
- Write comments in your code.
- If your code does not compile you will get `0`
- Do not share your code with your classmates.
- Do not print anything other than the expected output.
- You cannot use anything which is not covered in class.
- Do not submit any of the files you used for testing.

Turn in:

- Source code of a complete C program. Name of the file should be in this format: `<full_name>_PA4.c`.
- Example: `jane_doe_PA4.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>_PA4.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`.
- 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 asking question: `-30`.
- Fails at parsing leaf nodes: `-30`.
- The result is wrong: `-20`.
- Output format is wrong: `-30`. (Be careful with spacing)
- Infinite loop or recursion: `-90`.
- Prints anything extra: `-30`.
- Unwanted chars and spaces in the output: `-30`.
- Submission includes files other than the expected: `-10`.
- Submission does not follow the file naming convention: `-10`.
- Sharing or inheriting code: `-200`.
- IF YOU DON'T FOLLOW THE FILE NAMING CONVENTIONS YOU WILL GET `0`.

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.