COSC2430: Programming and Data Structures

Postfix calculator

Introduction

In this homework, you will create a C++ program that reads an infix expression from file, converts it to postfix notation, and computes the result.

Input and Output

The input is a single text file containing the infix expression for which you need to compute the result. All operands and operators are separated by spaces (the number of spaces may vary).

Example of input

25 + (4 * 5) – 12

The output should contain the expression in postfix notation and the result written in decimal form (2 digits after the comma).

Example of output

25 4 5 * 12 - +
33.00

Additional calculator properties

Besides the basic operations described above, the calculator should be able to perform the following tasks:

- **Parenthesis check**. The calculator should verify that the input file contains an equation with balanced brackets. In case of unbalanced equation, the output file should be empty. Curly brackets, square brackets and parenthesis are all acceptable. They do not need to appear in a particular order, as long as they are balanced
  Examples of unbalanced equations:
  5 + [12 * (3 + 7) //missing]
  5 + [12 * (3 + 7] -2) //wrong order

- **CE (clear everything).** The presence of this symbol clears the equation input so far and restarts from an empty stack.
  Example: 5 + [12 * (3 CE (4 * [5 + 7] – 2)
  Result: 4 5 7 + * 2 –
  46.00

- **Overwrite operators.** If the input file includes two valid operators consecutively, the latter overwrites the former.
  Example: 4 * + [5 + 7] – 2.5
  Result: 4 5 7 + 2.5 – +
  13.50

- **Missing operators.** If two operands appear consecutively and are not separated by an operator, the program should output an empty file.
  Example: 25 + (4 5) – 12

- **C (clear).** This symbol eliminates the last entry (operator or operand). Make sure that this does not result into an unbalanced expression.
  Example: 25 + (**4** C 2 * 5) – 12 //valid after deletion of 4
  Result: 25 2 5 * 12 - +
  23.00
  Example: 25 + **(** C 4 * 5) – 12 //invalid, unbalanced parenthesis
  Result: <empty file>

The main C++ program will become the executable to be tested by the TAs. The result should be written on another text file (output file), provided on the command line. The input and output files are specified in the command line, not inside the C++ code.

The general call to the executable (calculator, in this example) is as follows:

```
calculator "A=<file>;C=<file>"
```

Call example with one input file and another output file.

```
calculator "A=a.txt;C=c.out"
```

Requirements

- **It is NOT allowed to use vector classes or other classes provided in the STL.**
- You can use array stacks or linked stacks for this homework. The array stack should have an appropriate size for the input, avoiding unnecessary waste of memory. A stack is too large if it is never filled more than half its size.
- **You must delete all dynamically allocated memory.**
- Your C++ code must be clear, indented and commented.
- Your program will be tested with GNU C++. Therefore, you are encouraged to work on Unix, or at least make sure that your code compiles and runs successfully on the server.
- You can use other C++ compilers, but the TAs cannot provide support or test your programs with other compilers.
- The output file must contain the result in the format specified.

Testing for grading:

- All the cpp files will be automatically compiled. Make sure you have only 1 file containing main(). If there is an error in compilation it will not be executed. **Programs that do not compile on the server will receive a score of 0.** You are responsible of the content of your submission folder.
- **The name of your submission folder must be hw2**. The folder must not be nested in another folder. Make sure that the TAs have read access to your folder.
- **Submitted files must be limited to headers and source files (.h and .cpp).**
- Your program should not crash, halt unexpectedly, take an unusual amount of time to run (more than 10 seconds) or produce unhandled exceptions.
- Your program will be tested with 10 test cases, going from easy to difficult.

DEADLINE: March 18th, 11:59PM – NO LATE SUBMISSIONS

Grading

- A program not submitted by the deadline or that does not compile is worth zero points.
- A program that compiles but does not work is worth 10 points.
- A code with no comments will receive a 5 points penalty.
- A code with unnecessarily large stacks (see Requirements) will receive a 5 points penalty.
- **Bonus (10 pt): invalid symbols check.** The calculator should ignore invalid symbols in the input file, and produce an output if the equation cleared of the wrong symbols is valid and balanced. Valid symbols include: + - * / . ( ) { } [ ] C CE. Wrong symbols may disrupt the usual spacing in the input format.
  Example: 25 + (2.5#* 4) – 12
  Result: 25 2.5 4 * 12 - +
  23.00
- **A program with memory leaks (non-deleted dynamic memory) will receive a 20 points penalty.** Use the following command to test your executable for memory leaks:
  `valgrind --leak-check=yes ./calculator "A=<file>;C=<file>"`

Plagiarism and cheating: C++ code is individual: it cannot be shared (other than small fragments used in class or in lab examples). Programs will be compared for plagiarism. If the TAs detect that a portion of your C++ code is highly similar to C++ on the Internet or other student the grade will be decreased at least 50%.