Assignment 03
Matching Parenthesis and Printing Sub-Expression using Stacks and Linked Lists

DS286.Aug16 Data Structures and Programming

September 15, 2016

Submission is due on or before **Sunday, September 28, 2016, 11:59pm IST**.
The assignment carries **75 points**, which is 7.5% of the course weightage.

1 Question

Programming IDEs such as Eclipse highlight the expression that fall between a pair of matching brackets to aid the developer. This involves locating the position of the left and right brackets, and the expression/code that falls between them, and highlighting the text between these positions. This assignment will simulate this behavior for simple mathematical expressions provided as input.

This assignment requires you to check if a given arithmetic expression is *well-formed or not*, as defined below. Note that we will use the general term "bracket" to mean parenthesis (), curly braces {} or square brackets [], unless noted otherwise.

1. The expression only contains the following characters: three types of brackets `{, [, (, ), ], }`, the digits `0-9`, the four operators `+, -, *, /` or a whitespace ' '
2. The number of left and right brackets of each type should be equal.
3. For each right bracket in the expression, the closest preceding unmatched left bracket should be of the same type.

For example, the following are well-formed expressions:

```
15/[(2+3)-{4+5}-1]
[(1 * (2 + 3) - 5) + (3 + (4 - 5) * 3)]
```

Whereas, the following are not:

```
"3^5"
[([2+3)-4+5]
```

```
((1 * (2 + 3) - 5) + (3 + (4 - 5) * 3)]
```

The input expression may contain any printable character. You may assume that only non-negative digits will be provided in the input expression. You may also assume that the four binary operators will be used in a valid manner in the input. For e.g. you will not get expressions like `-35+5` or `(1 2 *)`.

For each pair of left and right bracket, you should print the expression that falls between the brackets, including the bracket. Whitespaces should be omitted. In addition, you should also print the position of the left and right bracket in the original input expression to the program.

In solving your assignment, you are also expected to implement a `LinkedList` data structure, and also a `Stack` and a `List` data structures using the `LinkedList`, that will be used in your solution. These data structure implementation themselves are expected to work correctly on their own, in addition to being used by the parenthesis matching program.

2 Program Outline

You are given 6 header files `LinkedList.h`, `Stack.h`, `List.h`, `Global.h`, `ParenMatch.h` and `Structures.h`, and one program file `main.cpp` with the outline. Your first task is to implement a linked list data structure in the file `LinkedList.cpp` with all methods present in `LinkedList.h`. Then you should implement the methods for the stack and the list data structures *using the linked list class*, in the files, `Stack.cpp` and `List.cpp`, respectively. As you can see, `c++ templating` is used by LinkedList, Stack and List, and this makes them generic data structures that can store any element type. Note that all the methods of in the LinkedList, Stack and List header files *must* be implemented even if only some of them are used in the parenthesis matching program.

The given main file `main.cpp` accepts a single arithmetic expression as a string of characters as the command line argument. Only one string expression will be given at a time as input. Then it parses this expression by calling the `toList()` method to create a list of `Item` objects for each bracket, integer and operator in the expression, with their corresponding type, position and value. You will need to implement the `toList()` method.

Then for each item in the list, one of these functions is called depending on the item's type as follows:

1. If the item type is a left bracket, it calls `handleLeft()`
2. If the item type is right bracket, it calls `handleRight()`
3. If the item type is an integer or operator, it calls `handleExpr()`

You need to implement these 3 methods as well. The functionality of the code should be such that:

1. It should print ‘‘`Invalid parse`’’ if the input string contains characters other than the three bracket types, integers, operators `+, -, *, /`, or a whitespace, and terminate.

2. As soon as two brackets are matched, it should print the expression between the two brackets, including the brackets. This should be preceded by the start and end positions of the brackets in the original input expression.

3. At the end, it should print whether the given expression is well-formed or not.

All four of these methods you implement for parenthesis matching will need to be present in the file `ParentMatch.cpp`. You must not modify `main.cpp` or any of the provided header files. You cannot use any additional global variables except the ones provided in `Global.h`.

You have to write the `Makefile` for this assignment yourself. It should produce an executable named `ParMatch.out` that corresponds to the `main.cpp` that is provided. Note the camel casing.

3 Sample inputs and results

```
./ParMatch.out "1+6^7"
Invalid parse

./ParMatch.out "[(1+2)*{3-6}/7])"
1,5,(1+2)
7,11,{3-6}
0,14,[(1+2)*{3-6}/7]
Given expression is not well formed :-(

./ParMatch.out "1-(2 + 3)+4"
2,8,(2+3)
Given expression is well formed :-)

./ParMatch.out "[(1*(2+3)-5)+(3+(4-5)*3)]"
4,8,(2+3)
1,11,(1*(2+3)-5)
16,20,(4-5)
13,23,(3+(4-5)*3)
0,24,[(1*(2+3)-5)+(3+(4-5)*3)]
Given expression is well formed :-)
```

4 Submission Instruction

Please follow these instructions carefully. We use automated scripts for evaluation. So a failure to follow these instructions will mean that your submission will not be evaluated.

- Only write your programs in the four files mentioned: `LinkedList.cpp`, `Stack.cpp`, `List.cpp` and `ParenMatch.cpp`. The program files should implement the methods from the corresponding header files.
- Do not modify the other files that are provided, namely `main.cpp` and the header files.
- Place all your files including source file, executable file and `Makefile`, in a single folder whose name is determined as follows. My full name is "Prateeksha Varshney" where "Prateeksha" is my first name, so the folder name should be `03Prateeksha` for my submission. Please note the capitalization of first letter of the first name. The final contents of the folder would be as follows:

```
03Prateeksha
|- Stack.h
|- Stack.cpp
|- LinkedList.h
|- LinkedList.cpp
|- main.cpp
|- ParenMatch.cpp
|- ParenMatch.h
|- Global.h
|- Structures.h
|- List.h
|- List.cpp
|- ParMatch.out
|- Makefile
```

- This folder should be compressed using the `tar` program as follows:
  `tar -cvf 03Prateeksha.tar 03Prateeksha/`
  Note: Any other compression format *will not be accepted* and will be treated as no submission. Its your responsibility to check if the file can be properly uncompressed and all files inside are intact.
- Send a separate mail to the TA's email address `prateeksha@grads.cds.iisc.ac.in` with the subject line `DS286.Aug16.A03`. Do not write anything more or less in the subject line. Do add any text in the body. Do not send the assignment as a reply-email to any other mail.

- **Only one submission will be accepted**. If multiple emails or files are received, **only the first one** will be taken as the submission. Only the submission received **before the deadline** will be accepted.
- Use only the C++ language for completing this assignment. Make sure the code compiles and executes correctly on the head node of the `turing.cds.iisc.ac.in` server using `g++` command. The code will be compiled and tested on this machine during evaluation.
- Indent/format the code and add inline comments describing that the code is supposed to do. This will help you debug better, and give us an insight on the logic you are using.
- It is your responsibility to remove all debug statements you may have added during development and testing your code. The evaluation your submission is done using automated scripts, and your console output will be tested against a predetermined correct output. If the outputs do not match exactly, it will be taken as wrong output.

5 Ethics

You should not get assistance from other students or external sources in directly solving the assignment. Getting help on generic C++ and data structures concepts, e.g., on using lists, strings, libraries, compilation, etc. is accepted. You are encouraged to post questions to the course mailing list so that the TA, instructors or other students can respond. This also ensures you do not have an unfair advantage/disadvantage over other students. If you have received assistance from other sources, send a *separate email to the Instructor and the TA* disclosing the external sources and type of support received.

By making a submission, you are asserting that all code that you submit was designed and developed by you. Do NOT copy and paste code from anyone else! All code will be verified using a plagiarism checker, and *penalties will be imposed* if plagiarism is found from unattributed sources.