CMPE 223
Fall 2021
Programming Homework 1

This assignment is due by 23:59 on Friday, 5 November 2021.

You are welcome to ask your HW related questions. You should use only one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:
   - CMPE223-HW1-OfficeHour1: 25 Oct, 06:00-07:00 PM, Zoom ID: 944 4772 7848
   - CMPE223-HW1-OfficeHour2: 30 Oct, 06:00-07:00 PM, Zoom ID: 967 4755 3443

Note: Please make sure that you have read the HW document well before participating. However, no HW related questions will be accepted except from the above options.

PROGRAMMING TASK(s)

Indiana Jones is a fictional professor of archaeology and begins his new journey to the ancient Egypt to unravel their secrets. He is stuck in front of a great gate of the pyramid with some arithmetic expressions and strings that are written on both halves of it. He believes that there could be a great treasure behind. To unlock that gate, it is required to solve these mysteries so that he calls for your help.

Task 1

First half of the gate needs to be decided if a given arithmetic expression is *well-formed* or *not*, as described below. For this task, you must use your own implementation of stacks by taking inspiration from your textbook. The stack should use generics and use a linked-list based implementation. You are not allowed to use any external library or .jar file.

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

As an example, while the expression `[(1 * (2 + 3) - 5) + (3 + (4 - 5) * 3)]` is well-formed, the expression `[([2+3)-4+5]` is not. The input expression can include any printable character. However, you can assume non-negative digits and four binary operators will be always used in a proper form. Here is the `sampleinput1.txt` file content:

```
3
[(1*(2+3)-5)+(3+(4-5)*3)]
[(1+2)*{3-6}/7])
1+6^7
```

First line of the input file shows the number of arithmetic expressions. 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. More formally, here are the output specifications:

- It should print "`Invalid Parse`" if the input string contains characters other than the three bracket types, integers, operators `+`, `-`, `*`, `/`, or a whitespace, and terminate.
- 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.
- At the end, it should print whether the given expression is well-formed or not.

The output for the `sampleinput1.txt` file is given as follows. Please check your program with this input file as well as the others that you will create. Please note that we may use other input files when grading your assignments.

```
% java Task1 sampleinput1.txt
Parsing expression #1:
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.
Parsing expression #2:
1,5,(1+2)
7,11,{3-6}
0,14,[(1+2)*{3-6}/7]
Given expression is not well formed.
Parsing expression #3:
Invalid Parse
```

Task 2

Second half of the gate needs to be decided if a given string is a *good string* and its *good value*, as described below. For this task, you must use your own implementation of queue by taking inspiration from your textbook. The queue should use generics and use a linked-list based implementation. You are not allowed to use any external library or .jar file.

For a string *T*, *T* is a *good string* if it contains all 26 of the lowercase English alphabets. For example, "abcdefghijklmnopqrstuvwxyz", "ababbcdefghijklmnopqrstuvwxyz" and "zyxwvut-srqponmlkjihgfedcbaxx" are all good strings; however, "azsd" and "abcdefghijklmnopqrstuvwxy" are not.

The *good value* of a string *S* is defined as the number of distinct integer pairs (i, j) with $i, j \geq 0$ and $i + j < |S|$ such that if we remove the first i character(s) and the last j character(s) from *S*, the resulted string *S'* is still a good string. (|*S*| is the length of string *S*.)

Here is the `sampleinput2.txt` file content:

```
2
ababbcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
```

First line of the input file shows the number of given strings to evaluate. It is guaranteed that each string *S* is not empty and consists of lowercase letters only. For each string to evaluate, you should output its good value in one line.
The output for the `sampleinput2.txt` file is given as follows. Please check your program with this input file as well as the others that you will create. Please note that we may use other input files when grading your assignments.

```
% java Task2 sampleinput2.txt
Parsing string #1:
3
Parsing string #2:
378
```

WHAT TO HAND IN

A zip file for both parts containing:

- The Java sources for your program.
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.

- You can test your Java source files on (if) available Moodle VPL environment to ensure your solution's correctness before submitting. VPL simply tests your program's output by checking against given sample input.
- A **maximum-3 pages** PDF report document that explains your own answers for programming task in a clearly readable PA report format (refer to **PA REPORT FORMAT** section).

PA REPORT FORMAT

A programming assignment report is a self-description of a programming assignment and your solution. The report must not be hand-written. You may use a word processor or the on-line editor of your choice and prepare as a PDF document. The report must be grammatically correct and use complete English sentences. Each report should include the following sections, in the order given:

Information (%5): This section includes your ID, name, section, assignment number information properly.

Problem Statement and Code Design (%30): Include a brief summary of the problem and/or your sub-tasks to be completed in this assignment. You should show your modular design rationale by creating a structure chart that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart using available graphical tools like MS PowerPoint, SmartDraw etc.

Implementation and Functionality (%40): Since you have modular source code, you should describe each sub-module (program) in this section. Each sub-module should include names and types of any input/output parameters as well as the pseudocode algorithm that used for completing its task. By this way, you give meaning to each chart boxes from the previous section.

Testing (%15): You should provide a tester class that is able to identify key test points of your program. This class should be able to generate additional (apart from the given sample input/output) test data for the purpose of being clear on what aspects of the solution are being tested with each set. This section should also include a description of any program *bugs* that is, tests which has incorrect results. You should write these to describe your tests, summarize your results, and argue that they cover all types of program behavior.

Final Assessments (%10): In this final section, you should briefly answer the following questions:

- What were the trouble points in completing this assignment?
- Which parts were the most challenging for you?
- What did you like about the assignment? What did you learn from it?

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on Friday, November 5th.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload files and any additional files if you wrote additional classes in your solution as a single archive file (e.g., zip, rar).

3. The standard rules about late homework submissions apply (20 points will be deducted for each late day). Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. Your classes' name MUST BE as shown in the homework description.

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-----------------------------------------------------
// Title: Scheduler tester class
// Author: Name/Surname
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-----------------------------------------------------
```

9. Since your codes will be checked without your observation, you should report everything about your implementation. Add detailed comments to your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
 void setVariable(char varName, int varValue)
//-------------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
// Postcondition: The value of the variable is set.
//-------------------------------------------------------
{
      // Body of the function
}
```

10. Indentation, indentation, indentation...

11. This homework will be graded by your TA, İbrahim İleri. Thus, you may ask him your homework related questions through HW forum on Moodle course page. You are also welcome to ask your course instructor İsmail Bora Çelikkale for help.