Data Structures and Algorithms Lab

Lab 08 **Marks 05**

Instructions

Work on this lab individually. You can use your books, notes, handouts etc. but you are not allowed to borrow anything from your peer student.

Marking Criteria

Show your work to the instructor before leaving the lab to get some or full credit.

Problem Text

One of the tasks that compilers and interpreters must frequently perform is deciding whether some **pairs of expression delimiters are properly paired**, even if they are embedded multiple pairs deep. Consider the following *C++* expression.

$$a = \big(f[b] - (c + d)\big) \,/\, 2;$$

The compiler has to be able to determine which pairs of opening and closing parentheses, square braces, etc. go together and whether the whole expression is correctly delimited. A number of possible errors can occur because of **unpaired delimiters** or because of **improperly placed delimiters**. For instance, the expression below lacks a closing parenthesis.

$$a = (f[b] - (c + d) \,/\, 2;$$

The following expression is also invalid. There are the correct numbers of parenthesis and braces, but they are not correctly balanced. The first closing parenthesis does not match the most recent opening delimiter, a brace.

$$a = (f[b) - (c + d]) \,/\, 2;$$

A **stack** is extremely helpful in implementing solutions to this type of problem because of its **LIFO (Last-in, First-out)** behavior. A closing delimiter must correctly match the most recently encountered opening delimiter. This is handled by pushing opening delimiters onto a stack as they are encountered. When a closing delimiter is encountered, it should be possible to pop the matching opening delimiter off the stack. If it is determined that every closing delimiter had a matching opening delimiter, then the expression is valid.

```
bool delimitersOk( const string &expression )
```

Returns ***true*** if all the parentheses and braces in the string are **legally paired**. Otherwise, returns ***false***.

Your task is to write a program that takes input from file **input.txt** and displays **valid** or **invalid** for each correct expression and incorrect respectively.

The input file **(input.txt)** will contain input and is in exactly the following format: First line of the input file will contain the total number of expressions exists in the file and then each new expression exists on a new line. *None of the expressions contain any **spaces/blank-character**.*

Sample *(input.txt)*

6

$a = \big(f[b] - (c + d)\big) \,/\, 2;$

$a = (f[b] - (c + d) \,/\, 2;$

$a = (f[b) - (c + d]) \,/\, 2;$

$3 * (a + b)$

$f[3 * (a + b)]$

$a = f[b + 3$

Output

```
Valid
Invalid
Invalid
Valid
Valid
InValid
```

☺ ☺ ☺ **BEST OF LUCK** ☺ ☺ ☺