Due Friday, March 20, 11.55pm

In this project you will implement the Stack ADT to work with the BRIDGES classes and implement an application to evaluate postfix expressions. This project will be done in 3 parts.

1. You will use the linked list based implementation from the book to implement a stack(Figures 4.4, 4.18, and 4.20, from the text). Demonstrate your implementation on a stack of integers; performa a sequence of push and pops and output the stack contents (you will need to write a print() function within the Stack ADT). **Due March 11, 11.55pm.**

2. You will adapt this implemention from part 1 to use the BRIDGES SLelement class, instead of the Link class. Test your implementation in a similar way, except that *you will send the stack contents to the Bridges visualizer.* **Due March 15, 11.55pm**

3. You will write a function to implement a postfix expression evaluator. The pseudocode for this is as follows:

```
Initialize the stack. // stack is empty
for (each operator or operand in the input expression){
    if (an operand)
        "push" into stack
    else if (an operator)
    {
        opnd2 = "pop" stack
        opnd1 = "pop" stack
        result = Evaluate opnd1 "oper"  opnd2
        "push" result onto stack
    }
}
result = "pop" stack
```

   Due March 22, 11.55pm

Implementation Details.

- **Input.** Several postfix expressions will be provided (strings). You will be provided input expressions in symbolic form (using the letters 'a' through 'h', for instance, 'a-b*c/d' will be provided as 'abc*d/-'

- The characters, 'a' through 'h' will represent symbols with values provided in the data file. You will read and parse this string.

- **Output.** Use the BRIDGES visualizer to trace the contents of the stack as the expression is evaluated. In other words, call the visualizer after each of push() and pop() operations and the list contents will go to a unique web page for review.

- Since the operands are in symbolic form, you will need a mechanism to look up the values of symbols. Use the Java HashMap, which is essentially a lookup table based on key-value pairs. For help, check the following links: `http://www.tutorialspoint.com/java/java_`

`hashmap_class.htm` and documentation `http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html` Thus, when an operand is encounterd, its value is looked up from the hash map for use by the evaluator.

Evaluation.

By interactive demo.