Assignment 5: Stacks Again!

First, read about the infix, prefix (Polish) and postfix (reverse Polish) notation for arithmetic expressions. Following are some representative links, there are tons of tutorials and programming examples:

- `https://en.wikipedia.org/wiki/Reverse_Polish_notation`
- `https://uomustansiriyah.edu.iq/media/lectures/5/5_2017_10_06!09_58_23_PM.pdf`
- `http://www.cs.man.ac.uk/~pjj/cs212/fix.html`
- Programs: `https://www.geeksforgeeks.org/stack-set-4-evaluation-postfix-expression/`
- Homeworks :-)) : `http://www.cs.csi.cuny.edu/~zelikovi/csc326/data/assignment5.htm`
- Even an online tool: `https://www.mathblog.dk/tools/infix-postfix-converter/`

Write a C program, using stacks (linked list implementation) that does the following: Given an infix expression that involves

- real operands
- the following binary operators: $*, +, -, /,$ or ˆ (where ˆ is the exponentiation operator)
- the following bracket symbols: $[, \{, (,), \},]$

a) checks if it is a valid infix expression b) converts it to postfix and c) evaluates the postfix expression. A valid expression means valid in the usual sense: balanced parentheses and correct use of binary operators. For example, {(}) is invalid, and so is 3+/2.

E.g. on input [3+(4*2)-7]*(2*3), you should output:

```
Valid
3 4 2 * + 7- 2 3 * *
24
```

On input [3+(4*2)]-7[(2*3), you should output

```
Invalid input: unmatched "[" found at symbol number 10.
```