Assignment # 4

February 7, 2009

This assignment brings us to a crucial aspect of programming called modular programming. Below is an excerpt from wikipedia on modular design.

> In systems/software engineering, modular design or "modularity in design" is an approach that subdivides a system into smaller parts (modules) that can be independently created and then used in different systems to drive multiple functionalities. Besides reduction in cost (due to lesser customization, and less learning time), and flexibility in design, modularity offers other benefits such as augmentation (adding new solution by merely plugging in a new module), and exclusion. Examples of modular systems are cars, computers and high rise buildings. Computers were the first systems in which modularity in architecture was implemented to overcome changing customer demands and to make the manufacturing process more adaptive to change (see modular programming). Modular design is an attempt to combine the advantages of standardization (high volume normally equals low manufacturing costs) with those of customization. Ref: `http://en.wikipedia.org/wiki/Modular_design`

Hence it's important to solve problems in a modular way. Now let us turn our attention to solve the problem # 5 of assignment # 3. We can easily see that the problem can be broken into several modules as follows. In this assignment each of thees modules as a seperate problem.

1. Identifying whether the given expression is infix, or prefix or posfix
2. Evaluating a postfix expression
3. Evaluating a prefix expression
4. Converting infix to postfix
5. Converting infix to prefix
6. Evaluating an infix expression
7. Converting prefix to infix
8. Converting prefix to postfix
9. Converting postfix to infix
10. Converting postfix to prefix

11. Removing unnecessary braces from infix expression

As we can see some modules presented above can make use of previously developed modules.

1 Identify

1.1 Description

Identifying whether the given expression is infix, or prefix or posfix

1.2 Input Format:

A single integer N followed by N test cases each on a single line.

1.3 Output Format:

Ouput *Infix* or *Prefix* or *Postfix* corresponding to each case

1.4 Sample Input

```
3
a+b
+ab
ab+
```

1.5 Sample Output:

```
Infix
Prefix
Postfix
```

2 Eval Postfix

2.1 Description

Evaluate a given Postfix expression.

2.2 Input Format:

A single integer N followed by N test cases each on a single line.

2.3 Output Format:

Output the value of the postfix expression.

2.4 Sample Input

```
2
35+
00+
```

2.5 Sample Output:

```
8
0
```

3 Eval Prefix

3.1 Description

Evaluate a given Prefix expression.

3.2 Input Format:

A single integer N followed by N test cases each on a single line.

3.3 Output Format:

Output the value of the prefix expression.

3.4 Sample Input

```
2
+35
-00
```

3.5 Sample Output:

```
8
0
```

4 Infix to Postfix

4.1 Description

Convert the given infix expression to postfix expression.

4.2 Input Format:

A single integer N followed by N test cases each on a single line.

4.3 Output Format:

as described earlier

4.4 Sample Input

```
2
3+5
0-0
```

4.5 Sample Output:

```
35+
00-
```

5 Infix to Prefix

5.1 Description

Convert the given Infix expression to Prefix expression.

5.2 Input Format:

A single integer N followed by N test cases each on a single line.

5.3 Output Format:

as described earlier

5.4 Sample Input

```
2
3+5
0-0
```

5.5 Sample Output:

```
+35
-00
```

6 Eval Infix

6.1 Description

Evaluate a given Infix expression.

6.2 Input Format:

A single integer N followed by N test cases each on a single line.

6.3 Output Format:

Output the value of the Infix expression.

6.4 Sample Input

```
2
3+5
0-0
```

6.5 Sample Output:

```
8
0
```

7 Prefix to Infix

7.1 Description

Convert the given Prefix expression to Infix expression.

7.2 Input Format:

A single integer N followed by N test cases each on a single line.

7.3 Output Format:

as described earlier

7.4 Sample Input

```
2
+35
-00
```

7.5 Sample Output:

```
3+5
0-0
```

8 Prefix to Postfix

8.1 Description

Convert the given Prefix expression to Postfix expression.

8.2 Input Format:

A single integer N followed by N test cases each on a single line.

8.3 Output Format:

as described earlier

8.4 Sample Input

```
2
+35
-00
```

8.5 Sample Output:

```
35+
00-
```

9 Postfix to Infix

9.1 Description

Convert the given Postfix expression to Infix expression.

9.2 Input Format:

A single integer N followed by N test cases each on a single line.

9.3 Output Format:

as described earlier

9.4 Sample Input

```
2
35+
00-
```

9.5 Sample Output:

```
3+5
0-0
```

10 Postfix to Prefix

10.1 Description

Convert the given Postfix expression to Prefix expression.

10.2 Input Format:

A single integer N followed by N test cases each on a single line.

10.3 Output Format:

as described earlier

10.4 Sample Input

```
2
35+
00-
```

10.5 Sample Output:

```
+35
-00
```

11 Braces removal

11.1 Description

Given an infix expression remove unnecessary braces from it and output the reduced infix expression.

11.2 Input Format:

A single integer N followed by N test cases each on a single line.

11.3 Output Format:

as described earlier

11.4 Sample Input

```
2
3+(5*5)
(((0-0)))
```

11.5 Sample Output:

```
3+5*5
0-0
```

12 Matrix walk

12.1 Description

You are given an $M \times N$ matrix whose elements are either 0 or 1. Infact, most of the elements in this matrix are 1. You are given the indices of an element (p,q) which is 1. Find the nearest 0 to it. Assume that from an element all its 8 neighbours (including diagonal) are at the same distance. Use the appropriate data structure. You are allowed to change the values of the matrix if needed.

12.2 Input Format:

M N Next M Lines: The matrix represented by 0/1's Final Line: Point of interest, p q

12.3 Output Format:

as described

12.4 Sample Input

```
54 1111 1111 1111 0111 1101 12
```

12.5 Sample Output:

```
3 0
```

13 Pouring water

13.1 Description

Given two vessels, one of which can accommodate a litres of water and the other - b litres of water, determine the number of steps required to obtain exactly c litres of water in one of the vessels.

At the beginning both vessels are empty. The following operations are counted as 'steps':

- emptying a vessel
- filling a vessel
- pouring water from one vessel to the other, without spilling, until one of the vessels is either full or empty

13.2 Input Format:

An integer t, $1 \leq t \leq 100$, denoting the number of testcases, followed by t sets of input data, each consisting of three positive integers a, b, c not larger than 40000, given in separate lines.

13.3 Output Format:

For each set of input data, output the minimum number of steps required to obtain c litres, or -1 if this is impossible.

13.4 Sample Input

```
2
5
2
3
2
3
4
```

13.5 Sample Output:

```
2
-1
```

Taken from `http://www.spoj.pl/problems/POUR1/`