ADS2 Mid-term CW

The aim of the coursework is to design and build an arithmetical calculator. The calculator will be coded in ‘Postfix++’, a simple language designed for the evaluation of postfix arithmetical expressions. You are asked to report on the design of your calculator (algorithms and data structures) and provide Javascript code of your implementation and a link to a five-minute video of your calculator. Please read the following carefully and then enter your responses in the text boxes on the Coursera mid-term coursework submission page.

Postfix++

Postfix++ is a stack-based computer language directed at the evaluation of arithmetic expressions. You will implement a Postfix++ interpreter that can evaluate Postfix++ code line-by-line, as entered, for example, on a mobile device.

Postfix arithmetic

Operators, in postfix arithmetical expressions, follow operands. For example, 3 4 + means 3 + 4. The postfix expression 3 4 5 + * is evaluated as follows:

3 4 5 + *
3 9 * (replace 4 5 + with the result of adding 4 to 5)
27 (replace 3 9 * with 27)

Postfix expressions are conveniently evaluated using a stack. An expression consisting of operands and operators (collectively, ‘tokens’), is read from left to right. Successive operands are pushed on a stack until an operator arrives. The appropriate number of operands are then popped from the stack, combined with the operator, and the result is pushed back on the stack. The result of a calculation is always to be found at the top of the stack. A stack is notated [a b c…] in the following example; the stack top is the leftmost token.

Tokens	Stack before	Action	Stack after
3 4 5 + *	[]	Read 3	[3]
4 5 + *	[3]	Read 4	[4 3]
5 + *	[4 3]	Read 5	[5 4 3]
+ *	[5 4 3]	Pop twice, evaluate, and push	[9 3]
*		Pop two twice, evaluate, and push	[27]

Postfix with variables

The ‘++’ refers to an enhanced form of postfix: a postfix with variables. Expressions can contain variable names; the value of the variable is used for the calculation. The assignment operator ‘=’ assigns a value to a variable. For example

A 3 =

will set the value of A to 3. It is equivalent to the infix assignment statement A = 3 that is to be found in many computer languages.

Tokens	Stack before	Action	Stack after
A 3 =	[]	Read A	[A]
3 =	[A]	Read 3	[3 A]
=	[3 A]	Pop twice and set the value of A to 3	[]

An interactive session might proceed as follows

```
> A 2 =
[]
> B 3 =
[]
>A B *
[6]
```

Symbol Table

A **symbol table** data structure associates keys with values. For example,

Example	Key	Value
Phone book	Name	Telephone number
DNS	URL	IP address
Education	Student ID	Module grades
Compiler	Variable name	Memory address
Dictionary	Word	Definition

A generic symbol table should support two operations, INSERT and SEARCH. There may also be a DELETE operation. No assumptions are made on the type and format of keys and values.

You will need a symbol table in your Postfix++ interpreter. The table will map variable name (key) to value.

The target hardware

A Postfix++ interpreter might run on a small device with very limited memory. This means that the variable namespace is small and consists only of the names ‘A’-‘Z’.