Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Systems Engineering Department - ENCS313
Linux Laboratory - 1st semester - 2015/16

Project #1
C-language under Linux
Due: December 15, 2015

Instructors: Dr. Aziz Qaroush, Dr. Hanna Bullata, Mr. Muawiya Asali

Problem: Stack interpreter

You are required to build an interpreter for a machine that has a single stack. The machine is primitive and thus it understands simple commands. Consider the following very primitive language for programming a stack machine:

Command	Meaning
int	push the integer *int* on the stack
+	push a '+' on the stack
s	push an 's' on the stack
e	evaluate the top of the stack (see below)
p	print content of the stack
d	delete the top of the stack
x	stop (exit the program)

Below is a brief description for each command in addition to an example in each case (the symbol > in the below examples refers to the prompt where the interpreter receives the commands):

- p: print content of the stack.

  Example

  ```
  > p
  ```

  The above command might print the following output (meaning the current content of the stack):

  ```
  110
  223
  +
  429
  -
  ```

- *int*: push the integer *int* on the stack.

  Example

  ```
  > 110
  ```

  The above will push the integer 110 on top of the stack.

- e: evaluate or execute a command. It's behavior depends on the top of the stack:
  - If + is on top of the stack, then the + is popped off the stack, the next two integers are popped and added, and the result is pushed back on the stack.
  - If s is on top of the stack, then s is popped off the stack and the next two items are swapped on the stack (thus the 2 elements remain on the stack).

- If d is on top of the stack, then d is popped off the stack and the current top of the stack is removed from the stack.
- If an integer is on top of the stack or the stack is empty, the stack is left unchanged.

The following examples show the effect of the e command in various situations; the top of the stack is on the left:

Stack before	Stack after
+ 1 2 5 s ...	3 5 s ...
s 1 + + 99 ...	+ 1 + 99 ...
1 + 3 ...	1 + 3 ...
d 1 2 5 s ...	2 5 s ...

You are required to implement the above interpreter as a singly-linked list. Input to the program is a series of commands, one command per line as shown above. Your interpreter should prompt for commands with the symbol >.

Assume that the stack deals only with unsigned integer numbers. Assume as well that the only allowed arithmetic command is +. In addition, assume that the allowed logical commands are & (AND), | (OR) and ∧ (XOR).

The interpreter should be able to handle errors if encountered. An example of an error you might get is when you're adding 2 popped elements from the stack, but one of the 2 elements is not an integer (e.g. + or &).

To do

- Write the code for the interpreter described above and name the executable as interpreter_single_stack. Generic functions must be located in separate C-files.
- Debug the application using the gdb debugger and/or the ddd interface.
- Use macros whenever necessary to add clarity.
- Make sure your code is clean and well indented, variables have meaningful names, etc.
- Make sure the C-files and header files have enough comments.
- Create a makefile that will help you compile the application.