Assignment 4 — Subroutines

Due: Wednesday, May 4th

While monolithic, straight-line code can theoretically be used to express any computation, code that incorporates some form of procedural organization is far easier to read, debug, and maintain. Subroutines allow common code to be grouped into callable units, so that it can easily be reused later.

Deliverables:

GitHub Classroom:	`https://classroom.github.com/a/YUZD-I26`
Required Files:	`stack.asm`, `balancer.asm`
Optional Files:	*none*

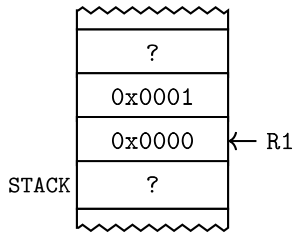
Part 1: Stacks

Recall that stacks can be implemented in assembly by allocating a contiguous block of memory and maintaining the *stack pointer*, the address of the top-of-stack, in a register.

Implement an assembly stack, growing from high addresses to low addresses, by completing the subroutines in `stack.asm`. The following code:

```
LEA R1, STACK
AND R2, R2, #0
JSR PUSH
ADD R2, R2, #1
JSR PUSH
JSR POP
```

...should place `0x0001` into `R2` while producing the following in memory:



Your subroutines must follow the LC-3 assembly conventions presented in lecture for saving and restoring any registers that they modify.

Part 2: Balancing Delimiters

In mathematical and technical writing, it is common to encounter delimiters that occur as complementary pairs, such as parentheses, curly braces, and square brackets. Such delimiters are said to be *balanced* if they are properly nested within one another. For example:

$$0 + [(1 + 2) \cdot 3] - \{4\}$$

The above expression is balanced, whereas, for example:

$$0 + [(1 + 2] \cdot 3) - \{4\}$$

The above expression is not balanced, as the right bracket ‘]’ appears before the right parenthesis ‘)’. That is to say, these delimiters exhibit LIFO behavior: the last delimiter opened must be the first delimiter closed, and the first delimiter opened must be the last delimiter closed.

Complete the assembly program in `balancer.asm`: this file contains the skeleton of a program to determine whether or not a given expression's delimiters are balanced by storing them on a stack. Code has also been included so that the subroutines in `stack.asm` can be called as though they were written in `balancer.asm`[1].

- You may *not* change any of the given code, however, you may add additional subroutines to the end of `balancer.asm`, if desired, according to the LC-3 conventions for saving and restoring registers.
- Your program must prompt the user to type an expression, which may be indefinitely long, and check that its parentheses, curly braces, and square brackets are balanced.
- If the user types an unbalanced closing delimiter, your program must immediately print an error message and quit, without waiting for the user to type any additional input.
- If the user hits the 'Enter' key, your program may assume that they will not type any additional input. It must ensure that there exist no unbalanced opening delimiters, then quit.
- It must be possible to rerun your program by manually resetting the PC to `0x3000`. It should not require that the LC-3 be reinitialized or that any files be reloaded.

You may assume that all input will consist of valid, printable ASCII characters and that the size of the stack should never need to exceed 16 elements.

For example:

```
Enter a string: 0 + [(1 + 2) * 3] - {4}
Delimiters are balanced.
```

Or, alternatively[2]:

```
Enter a string: 0 + [(1 + 2]
Delimiters are not balanced. Expected ')'.
```

Or, alternatively[3]:

```
Enter a string: 0 + 1 + 2)
Delimiters are not balanced. Expected '('.
```

Or, alternatively[2]:

```
Enter a string: 0 + [(1 + 2) * 3 - {4
Delimiters are not balanced. Expected '}'.
```

Your program will be tested using `diff`, so its printed output must match *exactly*.

Part 3: Submission

The following files are required and must be pushed to your GitHub Classroom repository by the deadline:

- `stack.asm` — Working implementations of assembly subroutines for stacks, as specified.
- `balancer.asm` — A working assembly program for balancing delimiters, as specified.

The following files are optional:

- *none*

Any files other than these will be ignored.

[1] Note that the LC-3 simulator allows multiple assembly files to be loaded, but each file sets the PC accordingly on load. Thus, in order to run this program, `balancer.asm` should be the final file assembled and loaded into the simulator.

[2] If a closing delimiter is encountered that does not match an earlier opening delimiter, or if an expression ends with outstanding opening delimiters, then your program's error message must include the expected closing delimiter.

[3] If a closing delimiter is encountered and there exist no outstanding opening delimiters, then your program's error message must include the expected opening delimiter.