10/10

```
.ORIG x3000
;
; Author: Zane Wonsey
; Date: 12/09/2013
;
; Side note: used subroutines given in book
;
; Subroutines for carrying out the PUSH and POP functions. This
; program works with a stack consisting of memory locations x3FFF
; (BASE) through x3FFB (MAX). R6 is the stack pointer
;
                                LD R6, POINTER

INPUT                   GETC
                                JSR PUSH

                                GETC
                                ADD R1, R0, 0
                                GETC
                                JSR DPUSH

                                GETC
                                JSR PUSH

; -----

                                JSR PEEK
                                OUT

                                JSR DPOP
                                OUT
                                ADD R0, R1, 0
                                OUT

                                JSR DPOP
                                OUT
                                ADD R0, R1, 0
                                OUT


                                BR ENDPROGRAM


RETURN_HERE             .fill 0

PEEK                    ST      R2, Save2       ; Saves registers that
                                ST      R1, Save1       ; are needed by POP.
                                LD      R1, BASE        ; BASE contains -x3FF
                                ADD     R1, R1, #-1     ; R1 contains -x4000
                                ADD     R2, R6, R1      ; Compare stack point
                                BRz     fail_exit       ; Branch if stack is
                                LDR     R0, R6, #0      ; The actual "pop"
                                BR      success_exit

DPUSH                   ST      R2, Save2       ; Saves registers that
                                ST      R0, Save1       ; are needed by PUSH.
                                LD      R0, MAX         ; MAX contains -x3FFB
                                ADD R2, R6, R0  ; Compare stack pointer to -x
```

```
                        BRz     fail_exit       ; Branch if stack is

                        ADD     R6, R6, #-1     ; Adjust the stack po
                        STR     R1, R6, #0      ; The actual "push"
                        LD R0, Save1
                        ADD R2, R6, R1
                        BRz fail_exit

                        ADD     R6, R6, #-1     ; Adjust the stack po
                        STR     R0, R6, #0      ; The actual "push"
                        BR success_exit

DPOP            ST      R1, Save1       ; Saves registers that
                        ST      R2, Save2       ; are needed by POP.
                        LD      R2, BASE        ; BASE contains -x3FF
                        ADD     R2, R2, #-1     ; R1 contains -x4000
                        ADD     R3, R6, R2      ; Compare stack point
                        BRz     fail_exit       ; Branch if stack is

                        LDR     R0, R6, #0      ; The actual "pop"
                        ADD     R6, R6, #1      ; Adjust stack pointe
                        ADD     R3, R6, R2      ; Compare stack point
                        BRz     fail_exit       ; Branch if stack is

                        LDR     R1, R6, #0      ; The actual "pop"
                        ST      R1, Save1
                        ADD     R6, R6, #1      ; Adjust stack pointe
                        BR      success_exit

POP                     ST      R2, Save2       ; Saves registers tha
                        ST      R1, Save1       ; are needed by POP.
                        LD      R1, BASE        ; BASE contains -x3FF
                        ADD     R1, R1, #-1     ; R1 contains -x4000
                        ADD     R2, R6, R1      ; Compare stack point
                        BRz     fail_exit       ; Branch if stack is
                        LDR     R0, R6, #0      ; The actual "pop"
                        ADD     R6, R6, #1      ; Adjust stack pointe
                        BR      success_exit

PUSH            ST      R2, Save2       ; Saves registers that
                        ST      R1, Save1       ; are needed by PUSH.
                        LD      R1, MAX         ; MAX contains -x3FFB
                        ADD R2, R6, R1  ; Compare stack pointer to -x
                        BRz     fail_exit       ; Branch if stack is
                        ADD     R6, R6, #-1     ; Adjust the stack po
                        STR     R0, R6, #0      ; The actual "push"

success_exit    LD      R1, Save1       ; Restore original
                        LD      R2, Save2       ; register values.
                        AND     R5, R5, #0      ; R5 <-- success
                        RET

fail_exit               LD      R1, Save1       ; Restore original
                        LD      R2, Save2       ; register values.
                        AND     R5, R5, #0      ;
                        ADD     R5, R5, #1      ; R5 <-- failure
                        RET
```

```
ENDPROGRAM              HALT

BASE                    .FILL    xC001   ; BASE contains -x3FFF.
MAX                              .FILL    xC005
POINTER                 .FILL    x3FFF
Save1                   .FILL    x0000
Save2                   .FILL    x0000

.END
```