Lab3 Report

Algorithm Explanation

First of all, initial the flexible queue(stack) pointer. We have head pointer and tail pointer, and also the outs pointer:

```
        LD R5,HEAD  ;;R5 <- x5000
        LD R6,TAIL  ;;R6 <- x5000
        LD R7,OUTS  ;;R7 <- x6000
        ...
HEAD    .FILL x5000
TAIL    .FILL x5000
OUTS    .FILL x6000
OUTSTA  .FILL x6001
```

Then we need to process the input. And according to the input, we branch to different subroutine:

```
GET-    LD R1,ASC-      ;;R1 <- -45
        ADD R1,R0,R1    ;;R1 <- R0 - 45
        BRz POPL
GET]    LD R1,ASC]      ;;R1 <- -93
        ADD R1,R0,R1    ;;R1 <- R0 - 93
        BRz POPR
GET+    LD R1,ASC+      ;;R1 <- -43
        ADD R1,R0,R1    ;;R1 <- R0 - 43
        BRz PUSHL
GET[    LD R1,ASC[      ;;R1 <- -91
        ADD R1,R0,R1    ;;R1 <- R0 - 91
        BRz PUSHR
```

Then we should use four subroutine : POPL, POPR, PUSHL, PUSHR. For example POPL, we should pop from head pointer to outs pointer, let head pointer + 1, let counter - 1 and let outs pointer + 1 :

```
POPL    ADD R4,R4,0     ;;setcc : R4
        BRz EMPTYPO
        ADD R2,R4,#-1   ;;R2 <- R4 - 1
        BRz ONEPOP
        ADD R7,R7,#1    ;;R7 <- R7 + 1 OUTS pointer ++
        LDR R2,R5,#0    ;;R2 <- MEM[R5]
        STR R2,R7,#0    ;;MEM[R7] <- R2
        ADD R5,R5,#1    ;;R5 <- R5 + 1 head pointer ++
        ADD R4,R4,#-1   ;;R4 <- R4 - 1 counter --
        BRnzp INPUT     ;;STEP DONE ,NEXT INPUT
```

For example PUSHR: we should let tail pointer + 1 , then push char into tail pointer:

```
PUSHR   TRAP x20    ;;GETC
        TRAP x21    ;;ECHO
        ADD R4,R4,0     ;;setcc : R4
        BRz EMPTYPU
        ADD R6,R6,#1    ;;R6 <- R6 + 1
        STR R0,R6,0     ;;MEM[R6] <- R0
        ADD R4,R4,#1    ;;R4 <- R4 + 1
        BRnzp INPUT
```

What's more, we should pay attention to the empty push and pop. Thus when counter(R4) == 0, We branch to the EMPTYPO or EMPTYPU : in these subroutine, we don't move the head or tail pointer:

```
EMPTYPO ADD R7,R7,#1    ;;R7 <- R7 + 1
        LD R2,ASC_      ;;R2 <- 95
        STR R2,R7,0     ;;MEM[R7] <- R2 ('_')
        BRnzp   INPUT

EMPTYPU STR R0,R6,0     ;;MEM[R6] <- R0
        ADD R4,R4,#1    ;;R4 <- R4 + 1
        BRnzp INPUT
```

Finally, when input is `ENTER` ,we branch to ENDIN: we puts in OUTSTA pointer.

```
ENDIN   AND R2,R2,0     ;;R2 <- 0
        ADD R7,R7,#1    ;;R7 <- R7 + 1
        STR R2,R7,0     ;;MEM[R7] <- 0
        LD  R0,OUTSTA   ;;R0 <- x6001
        TRAP x22        ;;PUTS
        HALT
```

Question & Answers

1. How do you judge the empty situation?

   Answer: Using counter R4. When R4 == 0, it is empty.

2. Where do you store your output list?

   Answer: Using outs pointer. And finally trap x22 at location x6001.