Report 3

Algorithm explanation

This algorithm contains 3 parts: initialization, input and output, push and pop.

- initialization:
  - load x4000 and x4001 as the head and tail of the queue, load x5000 as the place to store the output.
  - R6 is the ptr of queue_head, R7 is the ptr of the queue_tail. R1 is the head of output place, R2 is the ptr of the output queue.

```
;init
LD R6,queue_head
LD R7,queue_tail
LD R2,put
LD R1,put
queue_head .fill x4000
queue_tail .fill x4001
put     .fill x5000
```

- input an output
  - input

    `1.` getc and put it out

    `2.` if its ASCLL mode is 10, break to output part

    `3.` else if its ASCLL mode is equal to '+' / '-' / '[' / ']', break to 'push_l' , 'pop_l' , 'push_r' , 'pop_r'.
  - output

    `1.` if R1 equals to R2, there is nothing in the output queue, exit.

    `2.` else put out the thing in R1, R1++
- push and pop
  - push_l and push _r

    `1.` getc and out

    `2.` store in R6/R7 point place

    `3.` R6-- / R7++
  - pop_l and pop_r

    `1.` if R7-R6==1, then there is nothing in the queue, put '_' in the output queue.

    `2.` then R6++ / R7--, put the thing in the output queue.

Essential parts

input and judge (+)

```
;+
LD R3,ASCLL_+
NOT R3,R3
ADD R5,R0,R3
ADD R5,R5,#1
BRz push_l
```

output the queue

```
output  NOT R3,R2
        ADD R3,R3,R1
        ADD R3,R3,#1
        BRz EXIT
        LDR R0,R1,#0
        OUT
        ADD R1,R1,#1
        BR output
```

push the queue (left)

```
push_l  GETC
        OUT
        STR R0,R6,#0
        ADD R6,R6,#-1
        BR input
```

pop the queue (left)

```
pop_l
        NOT R5,R6
        ADD R5,R5,R7
        BRz op_
        BR pop_con
op_     LD R4,_
        STR R4,R2,#0
        ADD R2,R2,#1
        BR input
pop_con ADD R6,R6,#1
        LDR R4,R6,#0
        STR R4,R2,#0
        ADD R2,R2,#1
        BR input
```

Q&A

`1.` describe how you output the queue

Answer: I store the output queue in the x5000. Using R1 points x5000, and R2 points to the tail of the output queue.

if the output queue is none, halt the program, else put out the thing pointed by R1, R1++, until R1==R2 is true.

`2.` how you pop and push it

Answer: push: getchar and let it in the queue by R6 or R7, then R6-- or R7++ to insure their pointed place is the head and tail of the queue.

pop: judge whether the queue is none(R7-R6-1=0), if it is, put '_' in the output queue, else R6++ or R7--, then put the thing pointed by R6 or R7 to the output queue.