Lab 3 Report

1 Instruction

In this lab, we will implement a data structure called deque, which supports pop and push on both sides.

2 Solution

We will do this by making some changes on a queue. The differences are that the Rear Pointer and the Front Pointer in deque can both push and pop elements. We define push and pop on the left are operations on the Front Pointer, and push and pop on the right are operations on the Rear Pointer. During the input section, after getting the character, we check which character it is and then jump to the corresponding operating section. In the pop part, we save the character popped in a array in order. After the input is done, which is after we press `ENTER`, output the array from the first element in order.

2.1 Algorithm

```
procedure Creat_Deque
a[101] : integer// creat the deque
b[30] : innteger// save characters that to be output
*f <- a[50]// initialize the front pointer
*r <- a[50]// initialize the rear pointer
*p <- b// p points to the output array
while x <- getchar() != 0 do
    if x = "+"// push to the left side
        x <- getchar()
        *f <- x// push operation
        f <- f-1// update the front pointer
    else if x = "-"// pop from the left side
        if f = r// the deque is empty
            *p = "_"
            p <- p+1
            else// the deque is not empty
                f <- f+1
                *p = *f
                p <- p+1
        else if x = "["// push to the right side
            x <- getchar()
            r <- r+1
            *r <- x
            else if x = "]"// pop from the right side
                if f = r// the deque is empty
                    *p = "_"
                    p <- p+1
                else// the deque is not empty
                    *p <- *r
                    p <- p+1
                    r <- r-1
*p = 0// set the end of the output string
puts(b)// output the string
```

2.2 Essential part of the code

```
pul     TRAP x20; load the letter to be pushed
        TRAP x21; echo
        STR R0, R6, #0; store the input value into the deque
        ADD R6, R6, #-1; the Front pointer move one to the left
        BR ldp; read the next character

pol     NOT R1, R6
        ADD R1, R1, #1; get the inverse of R6
        ADD R1, R1, R7
        BRz EM; check if the deque is empty
        ADD R6, R6, #1; carry out the pop operation
        LDR R0, R6, #0; R0 gets the value poped
        STR R0, R3 ,#0; save the value popped to the output array
        ADD R3, R3, #1; update the output array pointer
        BR ldp; read the next character
```

The code above shows how to push and pop to the left side. In the push part, we simply store the value into the front and move the pointer one address left. In the pop part, we first check if the deque is empty. If it is, we load the character "_" in the output array. If it is not, we move the Front Pointer one address right and store the value of the address the pointer points to in the output array.

The push and pop to the right side part, which are operations on the Rear Pointer, are very similar to the operations on the Front Pointer.

3 Q & A

> Q: What if we push too many letters in the deque?

A: Stored letters may overwrite other commands in the memory, causing the program to crash.

> Q: How many letters can the deque you create store?

A: At most 50+50+1 = 101.