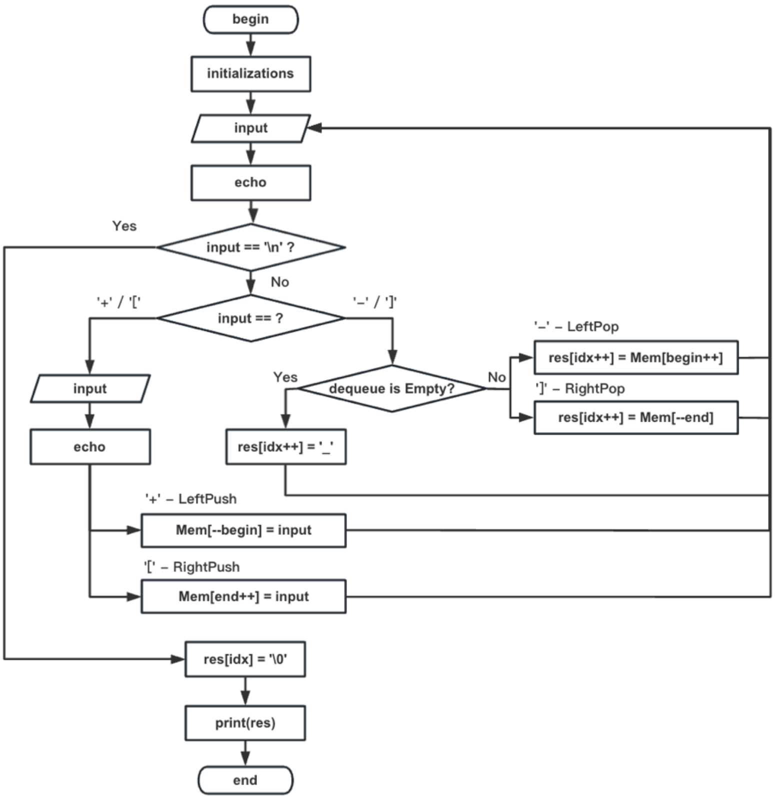
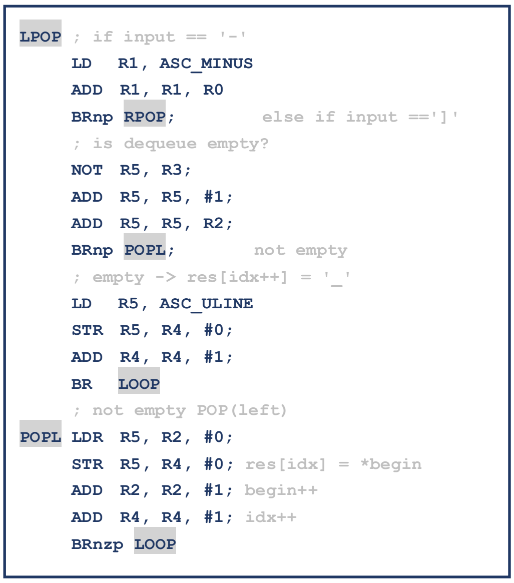
Report for LAB-3

1 Algorithm (flowchart)

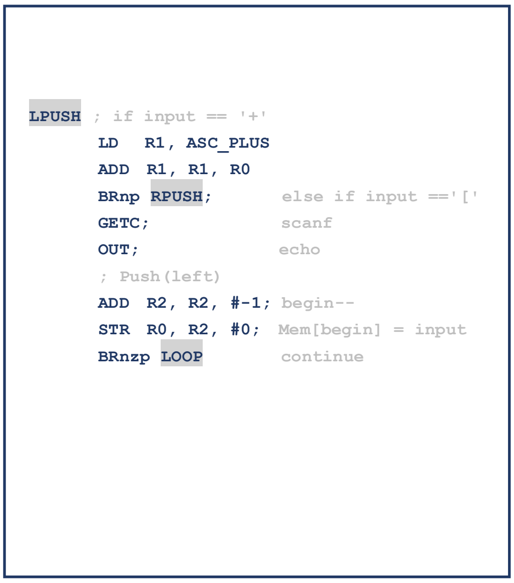


2 Code (essential parts with comments)

Take POP & PUSH operation on the left side for example:



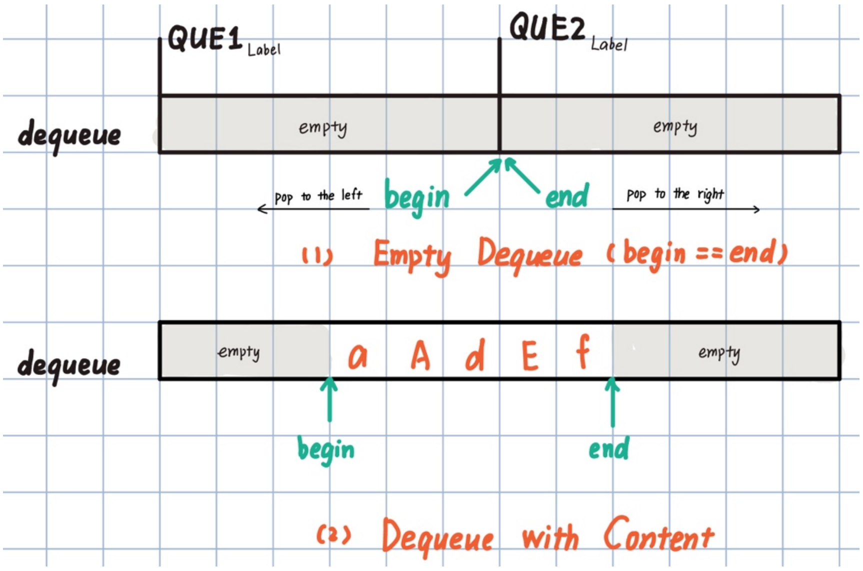
```
LPOP ; if input == '-'
     LD   R1, ASC_MINUS
     ADD  R1, R1, R0
     BRnp RPOP;          else if input ==']'
     ; is dequeue empty?
     NOT  R5, R3;
     ADD  R5, R5, #1;
     ADD  R5, R5, R2;
     BRnp POPL;         not empty
     ; empty -> res[idx++] = '_'
     LD   R5, ASC_ULINE
     STR  R5, R4, #0;
     ADD  R4, R4, #1;
     BR   LOOP
     ; not empty POP(left)
POPL LDR  R5, R2, #0;
     STR  R5, R4, #0; res[idx] = *begin
     ADD  R2, R2, #1; begin++
     ADD  R4, R4, #1; idx++
     BRnzp LOOP
```



```
LPUSH ; if input == '+'
      LD   R1, ASC_PLUS
      ADD  R1, R1, R0
      BRnp RPUSH;       else if input =='['
      GETC;             scanf
      OUT;              echo
      ; Push(left)
      ADD  R2, R2, #-1; begin--
      STR  R0, R2, #0;  Mem[begin] = input
      BRnzp LOOP        continue
```

3 Q & A

How you achieve this?



I use two .BLKW instruction (labeled by QUE1 & QUE2 respectively) to allocate the memory space for the dequeue, and use QUE2 to initialize the value of begin & end, making then both point to the center of the dequeue:

- begin points to the first element in the dequeue
- end points to the next location to be filled (to the right)

We can simply know whether the dequeue is empty by comparing begin == end

But execute POP/PUSH operations on each side would be different:

- For the left side:
  POP() means result.append(Mem[begin]); begin++;
  PUSH() means begin--; Mem[begin] = input;

- For the right side:
  POP() means end--; result.append(Mem[end]);
  PUSH() means Mem[end] = input; end++;