DAA

1002254987 Kutrodiya Neel HW18

10.1-1

	1	2	3	4	5	6
S	4					
	↑ top					

PUSH (S, 4)

	1	2	3	4	5	6
S	4	1				
		↑ top				

PUSH(S, 1)

	1	2	3	4	5	6
S	4	1	3			
			↑ top			

PUSH (S, 3)

	1	2	3	4	5	6
S	4	1				
		↑ top				

POP(S)

	1	2	3	4	5	6
S	4	1	8			
			↑ top			

PUSH(S, 8)

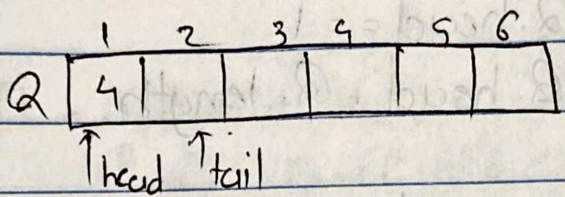
	1	2	3	4	5	6
S	4	1				

POP(S)

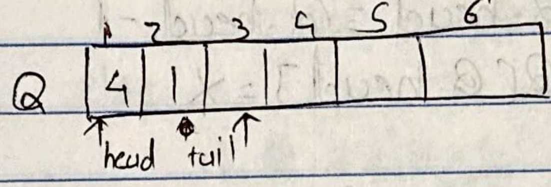
This is array obtained by performing the given operations On an empty stack with array S[1, ... 6]

(10.1-3) Given sequence.

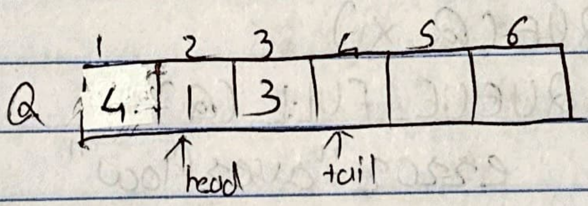
ENQUEUE(Q,4), ENQ(Q,1), ENQ(Q,1), DEQ(Q), ENQ(Q,8), DEQ(Q) with array Q[1,....,6]



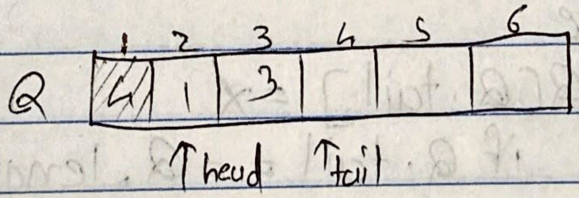
ENQ(Q,4)



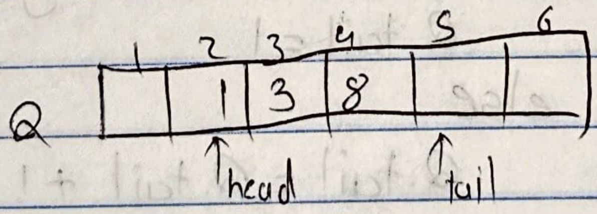
ENQ(Q,1)



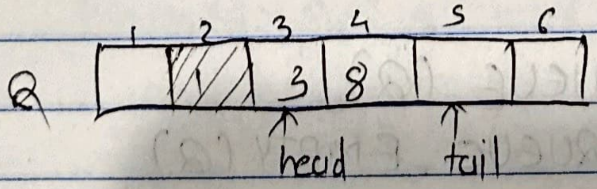
ENQ(Q,3)



DEQ.(Q)



ENQ(Q,8)



DEQ(Q)

∴ This is the final stack obtained after performing the given operations.

10.1-5

```
HEAD_ENQUE(Q,X)
    if QUEUE_full(z)
        enter "overflow"
    else
        if Q.head == 1
            Q.head = Q.length
        else
            Q.head = Q.head - 1
            Q[Q.head] = X

TAIL_ENQUE(Q,X)
    if QUEUE_FULL(Q)
        error "overflow"
    else
        Q[Q.tail] = X
        if Q.tail == Q.length
            Q.tail = 1
        else
            Q.tail = Q.tail + 1

HEAD_DEQUEUE(Q)
    if QUEUE_EMPTY(Q)
        error "underflow"
    else
        X = Q[Q.head]
        if Q.head == Q.length
        Q.head = 1
    else
        Q.head = Q.head + 1
        return x

TAIL-DEQUEUE (Q)
    if QUEUE-EMPTY x (Q)
            error "underflow"
    else
        if Q.tail == 1
            Q.tail = Q.tail - 1
            x = Q[Q.tail]
        return x
```

10.2-2

```
Stack_empty (L)
    if L.head =NIL
        return true
else
    return false
```

PUSH : adds an element in the begining of the list.

```
PUSH (L,x)
    x.next = L.head
    L.head = x
```

POP : removes first element from the list.

```
POP(L)
if stack.empty (L)
    error "underflow"
else
    x = L.head
    L.head = L.head.next
    return x
```

```
(10.4.3) Print_BinaryTree (T,S)
    PUSH (S, T.root)
      while ! stack_empty (s)
          x = s(S.top)
          while x! = NIL
              PUSH (s, x.left)
                x = s(s.top)
          POP(s)
            if ! stack_empty (s)
                x = POP(s)
                print x key
                PUSH(S, x.right)
```

Psuedo code:

```
inorder_traversal (root):
    stack = empty stack
     node = root
    while node is not null or stack is
        not empty;
      if node is not null:
            PUSH mode onto stack
            mode = node.left
    else:
        node = POP from stack
        print node.key
        node = node.right
```