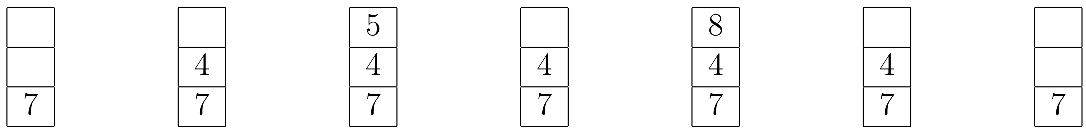
Data Structure and Algorithm Analysis(H)
Southern University of Science and Technology
Mengxuan Wu
12212006

Work Sheet 8

Mengxuan Wu

Question 8.1

1.

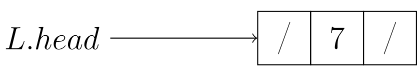


PUSH$(S, 7)$ PUSH$(S, 4)$ PUSH$(S, 5)$ POP(S) PUSH$(S, 8)$ POP(S) POP(S)

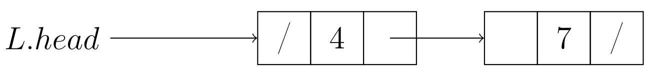
2.

7				ENQUEUE$(Q, 7)$
7	4			ENQUEUE$(Q, 4)$
7	4	5		ENQUEUE$(Q, 5)$
	4	5		DEQUEUE(Q)
	4	5	8	ENQUEUE$(Q, 8)$
		5	8	DEQUEUE(Q)
			8	DEQUEUE(Q)

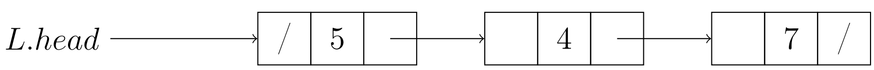
3.



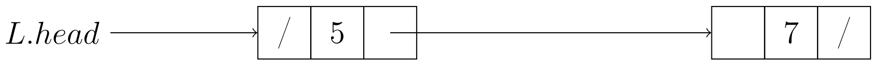
LIST-PREPEND$(L, 7)$



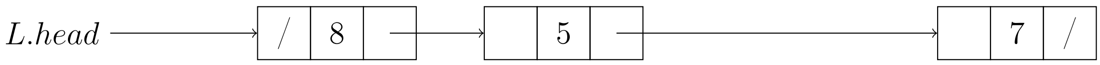
LIST-PREPEND$(L, 4)$



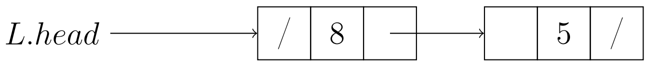
LIST-PREPEND$(L, 5)$



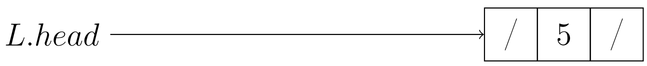
LIST-DELETE$(L, 4)$



LIST-PREPEND$(L, 8)$



LIST-DELETE$(L, 7)$



LIST-DELETE$(L, 8)$

Question 8.2

PUSH$S_1(A, x)$

if S_2.top - S_1.top == 1
    error "overflow"
else
    S_1.top = S_1.top + 1
    $A[S_1$.top$]$ = x

POP$S_1(A)$

if S_1.top == 0
    error "underflow"
else
    S_1.top = S_1.top - 1
    return $A[S_1$.top + 1$]$

Pseudo-code for S_1

PUSH$S_2(A, x)$

if S_2.top - S_1.top == 1
    error "overflow"
else
    S_2.top = S_2.top - 1
    $A[S_2$.top$]$ = x

POP$S_2(A)$

if S_2.top == $n + 1$
    error "underflow"
else
    S_2.top = S_2.top + 1
    return $A[S_2$.top - 1$]$

Pseudo-code for S_2

Question 8.3

```
ENQUEUE(Q, x)
if Q.head - Q.tail == 1 or (Q.head == 1 and Q.tail == Q.length)
    error "overflow"
else
    Q[Q.tail] = x
    if Q.tail == Q.length
        Q.tail = 1
    else
        Q.tail = Q.tail + 1
```

```
DEQUEUE(Q)
if Q.head == Q.tail
    error "underflow"
else
    x = Q[Q.head]
    if Q.head == Q.length
        Q.head = 1
    else
        Q.head = Q.head + 1
    return x
```

Question 8.4

Note: We assume the error handling is done by the stack.

Method 1:

```
ENQUEUE(Q, x)
while Stack-Empty(S2) == FALSE do
    PUSH(S1, POP(S2))
PUSH(S1, x)
```

```
DEQUEUE(Q)
while Stack-Empty(S1) == FALSE do
    PUSH(S2, POP(S1))
return POP(S2)
```

With this method, the runtime of these two operations depend largely on what is the last operation performed. If two same operations are performed consecutively, the latter one will be $\Theta(1)$. If two different operations are performed consecutively, the latter one will be $\Theta(n)$, where n is the number of elements currently in the queue.

Method 2:

```
ENQUEUE(Q, x)
PUSH(S1, x)
```

```
DEQUEUE(Q)
if Stack-Empty(S_2) == TRUE
    while Stack-Empty(S_1) == FALSE do
        PUSH(S_2, POP(S_1))
return POP(S_2)
```

In this method, the runtime of ENQUEUE is obviously $\Theta(1)$. The runtime of DEQUEUE is $\Theta(1)$ in average.

To explain the runtime, for every element in the queue, it will only be pushed into S_1 once, popped from S_1 once, pushed into S_2 once and popped from S_2 once. So the total runtime is $\Theta(4n) = \Theta(n)$. Since there are n DEQUEUE operations, the average runtime is $\Theta(1)$.