**Exercise 10.1**

1. **Explain how to implement 2 stacks in one array A[1:n] in such a way that neither stack overflows unless the total number of elements in both stacks is n.**

**Ans)**

Let S1 and S2 be the intern al stacks of array A with stack size n/2 each.

PUSH-ARRAY(A, x):

if A.S1.top == A.S1.size AND A.S2.top==A.S2.size:

error “stack overflow”

else

if A.S2.size == A.S2.top OR A.S2.top >= A.S1.top:

PUSH(A.S1 ,x)

else

PUSH(A.S2 , x)

POP-ARRAY(A)

if A.S1.top == 0 AND A.S2.top ==0:

error “underflow”

else:

if A.S1.top>= A.S2.top:

return POP(A.S1)

else

return POP(A.S2)

1. **Rewrite ENQUEUE and DEQUEUE to detect overflow and underflow.**

**Ans)**

//Tail is where next element will be inserted

ENQUEUE(Q, x):

if Q.head == Q.tail

error “overflow”

else:

Q[tail] = x

if Q.tail == Q.size:

Q.tail = 1

else

Q.tail = Q.tail + 1

DEQUEUE(Q):

if Q.head == Q.tail and Q.head ==1:

error “underflow”

else:

x = Q[head]

head = head + 1

if head == Q.size:

head = 1

return x

1. **Whereas a stack allows insertion and deletion of elements at only one end, and a queue allows insertion at one end and deletion at the other end, a *deque* (allows insertion and deletion at both ends. Write four -time procedures to insert elements into and delete elements from both ends of a deque implemented by an array.**

**Ans)**ENQUEUE-BACK(Q , x):

if Q.head == Q.tail:

error “overflow”

else:

Q[tail] = x

Q.tail = Q.tail + 1

if Q.tail > Q.size:

Q.tail = 1

ENQUEUE-FRONT(Q,x):

Q.head = Q.head -1

if Q.head == 0:

Q.head = Q.size

if Q.tail > 1:

if Q.head == Q.tail-1:

error “overflow”

if Q.tail == 1:

if Q.head == Q.size:

error “overflow”

else:

Q[head] = x

DEQUEUE-FRONT(Q):

if Q.head == Q.tail :

error “underflow”

else:

x = Q[Q.head]

Q.head = Q.head + 1

if Q.head > Q.size:

Q.head = 1

return x

DEQUEUE-BACK(Q):

if Q.head== Q.tail and. Q.head ==1:

error “underflow”

else:

Q.tail = Q.tail – 1

if Q[Q.tail] == 0:

Q.tail = Q.size

x = Q[tail]

return x

1. **Show how to implement a stack using 2 queues. Analyse the run time.**

**Ans)**

Push: Theta(1)

Pop: Theta(n)

PUSH(S, x):

if (S.Q1.head == 1 AND S.Q1.tail == S.Q1.size) or S.Q1.tail == S.Q1.head -1:

error “overflow”

else:

ENQUEUE(S.Q1, x)

POP(S):

if S.Q1.head == S.Q1.tail:

error “underflow”

else:

while S.Q1.head != S.Q1.tail – 1 OR !(S.Q1.head == S.Q1.size and S1.Q1.tail == 1):

ENQUEUE(S.Q2 , DEQUEUE(Q1)

x = DEQUEUE(Q1)

while S.Q2.head != S.Q2.tail:

ENQUEUE(S.Q1 , DEQUEUE(Q2)

return x