Data Structure
Chapter 2: stacks, queues

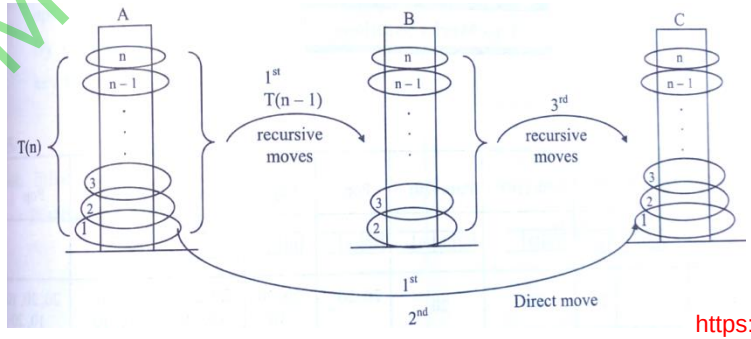
GATE CS PYQ
Solved By
Monalisa Pradhan

Q1 The recurrence relation capturing the optimal execution time of the *Towers of Hanoi* problem with n discs is

(A) $T(n) = 2T(n-2) + 2$
(B) $T(n) = 2T(n-1) + n$
(C) $T(n) = 2T(n/2) + 1$
(D) $T(n) = 2T(n-1) + 1$

GATE 2012,Q16,1 Mark:

- Let the three pegs be A, B and C.
- The goal is to move n pegs from A to C.
- To move n discs from peg A to peg C:
- move n-1 discs from A to B. This leaves disc n alone on peg A [n-1 move]
- move disc n from A to C [1 move]
- move n-1 discs from B to C so they sit on disc n [n-1 move]
- So the recurrence relation for Tower of Hanoi is
- T(n) = T(n-1)+1+T(n-1)
- T(n) = 2T(n-1)+1



- Ans : (D) T(n) = 2T(n-1)+1

Q2 Suppose a circular queue of capacity $(n-1)$ elements is implemented with an array of n elements. Assume that the insertion and deletion operations are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. The conditions to detect *queue full* and *queue empty* are

GATE 2012,Q35,2 Mark

(A) *full*: (REAR+1) mod n == FRONT
empty: REAR == FRONT

(B) *full*: (REAR+1) mod n == FRONT
empty: (FRONT+1) mod n == REAR

(C) *full*: REAR == FRONT
empty: (REAR+1) mod n == FRONT

(D) *full*: (FRONT+1) mod n == REAR
empty: REAR == FRONT

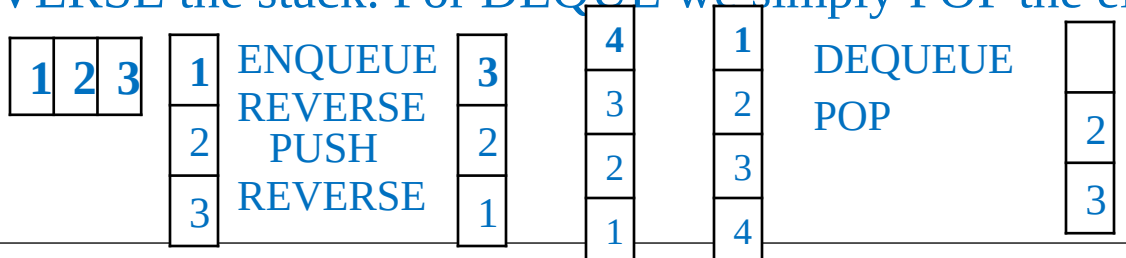
- *Queue full*: (REAR+1) mod n == FRONT
- *Queue empty*: REAR == FRONT
- Ans: (A)

- **GATE CS 2013 | Question: 44**
- Consider the following operation along with Enqueue and Dequeue operations on queues, where k is a global parameter.

```
MultiDequeue(Q)
{ m = k
  while (Q is not empty) and (m > 0)
 { Dequeue(Q)
  m = m – 1 } }
```

- What is the worst case time complexity of a sequence of queue operations on an initially empty queue?
- (A) $\Theta(n)$ (B) $\Theta(n+k)$ (C) $\Theta(nk)$ (D) $\Theta(n^2)$
- If after n enqueue we perform multiDequeue.
- 1. If k<n then one MultiDequeue run k times.
- 2. If n<k then one MultiDequeue run n times.
- So Worst case time complexity for MultiDequeue is $\Theta(n)$.
- Three possible operations :Enqueue, Dequeue and MultiDequeue.
- MultiDequeue is calling Dequeue k times.
- Since, the queue is initially empty, whatever be the order of these operations, there cannot be more number of Dequeue operations than Enqueue operations.
- Hence, the total number operations will be n only.
- Ans : (A) $\Theta(n)$

- GATE 2014 set-2,Q41,2 Mark: Suppose a stack implementation supports an instruction REVERSE, which reverses the order of elements on the stack, in addition to the PUSH and POP instructions. Which one of the following statements is TRUE *with respect to this modified stack*?
  (A) A queue cannot be implemented using this stack.
  (B) A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes a sequence of two instructions.
  (C) A queue can be implemented where ENQUEUE takes a sequence of three instructions and DEQUEUE takes a single instruction.
  (D) A queue can be implemented where both ENQUEUE and DEQUEUE take a single instruction each
- While ENQUEUE we REVERSE the stack, PUSH the element and then again REVERSE the stack. For DEQUE we simply POP the element.
- Ans : C



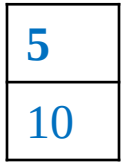
Q 4 The result evaluating the postfix expression 10 5 + 60 6 / * 8 − is

GATE 2015 Set-3,Q12,1 Mark

(A) 284 (B) 213 (C) 142 (D) 71

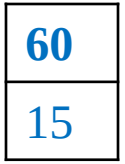
Push 10

Push 5

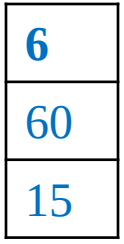


Pop top 2 element from stack do + and push result in stack

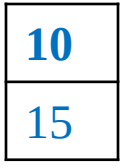
Push 60



Push 6

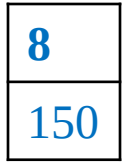


Pop top 2 element from stack do / and push result in stack



Pop top 2 element from stack do * and push result in stack

Push 8



Pop top 2 element from stack do - and push result in stack

Ans : (C) 142

Q 5 Consider the following recursive C function. GATE 2015 set-3,Q39,2 Mark

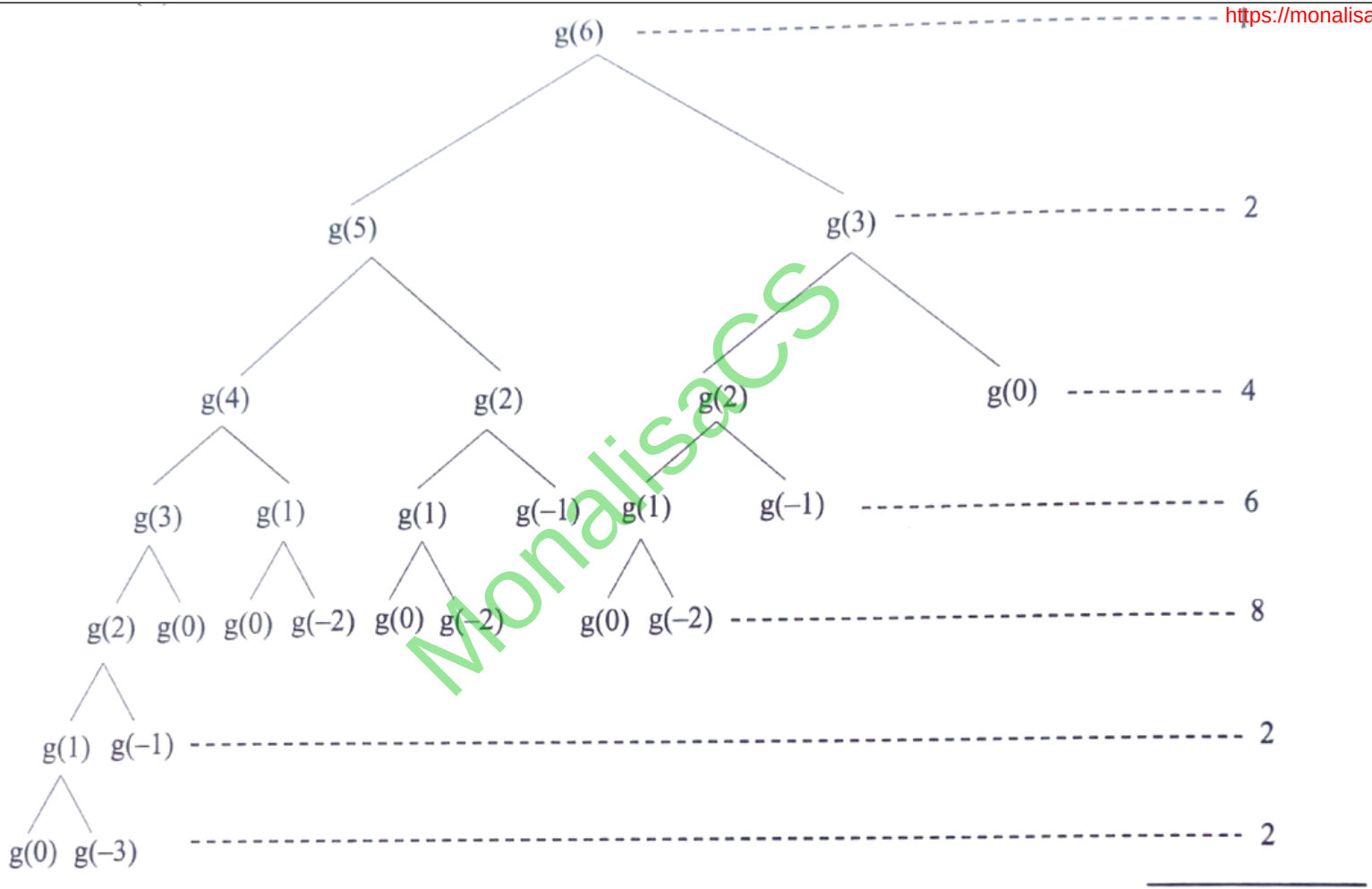
```
void get(int n)
 {
      if (n<1) return;
      get(n-1);
      get(n-3);
      printf("%d", n);
 }
```

If `get(6)` function is being called in `main()` then how many times will the `get()` function be invoked before returning to the `main()`?

(A) 15 (B) 25 (C) 35 (D) 45

- **T(n) = T(n-1) + T(n-3) +1 ;**
  T(n) = 1 ; for n < 1
- T(1)=T(0)+T(-2)+1=3 ,
- T(3) =T(2)+T(0)+1= 5 + 1 + 1 = 7 ,
- T(5) =T(4)+T(2)+1= 11 + 5 + 1 = 17,
- Ans : (B) 25

T(2)= T(1)+T(-1)+1=3 + 1 + 1 = 5,
T(4) = T(3)+T(1)+1= 7 + 3 + 1= 11 ,
T(6) = T(5) + T(3) + 1 = 17 + 7 + 1 = 25.



Q 6 A queue is implemented using an array such that ENQUEUE and DEQUEUE operations are performed efficiently. Which one of the following statements is **CORRECT** (n refers to the number of items in the queue)?

GATE 2016 set-1,Q10,1 Mark

(A) Both operations can be performed in $O(1)$ time

(B) At most one operation can be performed in $O(1)$ time but the worst case time for the other operation will be $\Omega(n)$

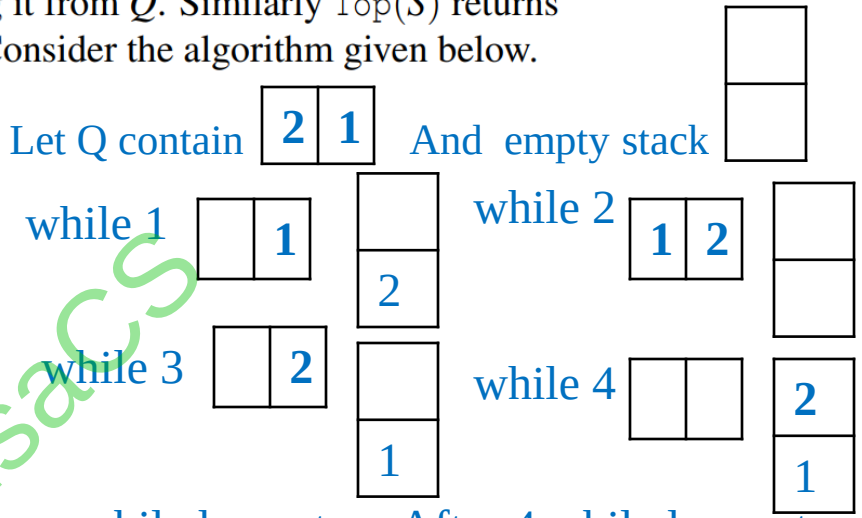
(C) The worst case time complexity for both operations will be $\Omega(n)$

(D) Worst case time complexity for both operations will be $\Omega(\log n)$

- Since it is mentioned in the question that both of the operations are performed efficiently.
- Hence even the worst case time complexity will be O(1) by the use of the Circular queue there won't be any need of shifting in the array.
- Ans : (A) Both operations can be performed in *O(1)* time

Q 7 Let Q denote a queue containing sixteen numbers and S be an empty stack. $\texttt{Head}(Q)$ returns the element at the head of the queue Q **without** removing it from Q. Similarly $\texttt{Top}(S)$ returns the element at the top of S **without** removing it from S. Consider the algorithm given below.

```
while Q is not Empty do
    if S is Empty OR Top(S) ≤ Head(Q) then
        x := Dequeue(Q);
        Push(S,x);
    else
        x := Pop(S);
        Enqueue(Q,x);
    end
end
```



- Q is empty so while loop stop ,After 4 while loop stop.
- For 2 number it run 2*2= 4 time,
- for 3 number it run 3*3=9 time

The maximum possible number of iterations of the **while** loop in the algorithm is _____. GATE 2016 set-1,Q41,2 Mark

- For 16 number it will run 16*16 time=256
- Ans:256

Q 8 The attributes of three arithmetic operators in some programming language are given below.

GATE 2016 set-1,Q45,2 Mark

Operator	Precedence	Associativity	Arity
+	High	Left	Binary
–	Medium	Right	Binary
*	Low	Left	Binary

The value of the expression **2 – 5 + 1 – 7 * 3** in this language is _____ .

- 2-5+1-7*3 [+ have highest precedence]
- =2-6-7*3 [- is right associatively so right – will operate first]
- =2-(-1)*3 [- has more precedence over *]
- =3*3 =9

- Ans : 9

Q9: A circular queue has been implemented using a singly linked list where each node consists of a value and a single pointer pointing to the next node. We maintain exactly two external pointers **FRONT** and **REAR** pointing to the front node and the rear node of the queue, respectively. Which of the following statements is/are CORRECT for such a circular queue, so that insertion and deletion operations can be performed in $O(1)$ time?

GATE 2017 Set-2,Q13,1 Mark

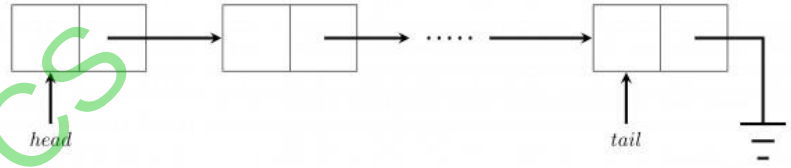
I. Next pointer of front node points to the rear node.
II. Next pointer of rear node points to the front node.

(A) I only
(B) II only
(C) Both I and II
(D) Neither I nor II

- Since, Circular queue deletes an item using Front pointer and insert an element using Rear pointer.
- If we want to insert an element into circular queue then we have to increment Rear pointer to next node then insert element.
- Then after update the next pointer of Rear node to the Front node.
- This method will have O(1) time for Insertion and Deletion.
- Only statement (ii) is true.
- Ans (B) II only

GATE CS 2018 | Question: 3

A queue is implemented using a non-circular singly linked list. The queue has a head pointer and a tail pointer, as shown in the figure. Let *n* denote the number of nodes in the queue. Let 'enqueue' be implemented by inserting a new node at the head, and 'dequeue' be implemented by deletion of a node from the tail.



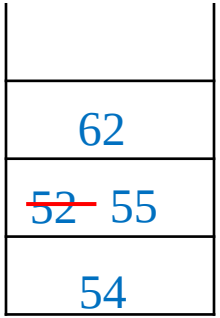
Which one of the following is the time complexity of the most time-efficient implementation of 'enqueue' and 'dequeue, respectively, for this data structure? (A)Θ(1),Θ(1) (B)Θ(1),Θ(n) (C)Θ(n),Θ(1) (D)Θ(n),Θ(n)

```
Enqueue( )
{ P→Data=Data
P → Next=Head
Head=P }
```

```
Dequeue()
 { temp=head
    While(temp→Next→Next!=NULL)
       temp=temp→next
     temp→next=NULL
     tail=temp}
```

- Enqueue, performs in constant time Θ(1), as it modifies only two pointers.
- We are traversing entire linked list for each Dequeue, so time complexity is Θ(n).
- Ans : (B)Θ(1),Θ(n)

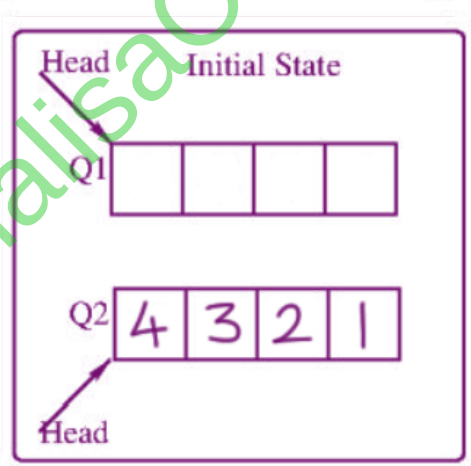
- GATE CS 2021 Set-1,Q21: Consider the following sequence of operations on an empty stack. Push(54);push(52);pop();push(55);push(62);s=pop();
- Consider the following sequence of operations on an empty queue.
- enqueue(21);enqueue(24);dequeue();enqueue(28);enqueue(32);q=dequeue();
- The value of s+q is _____.

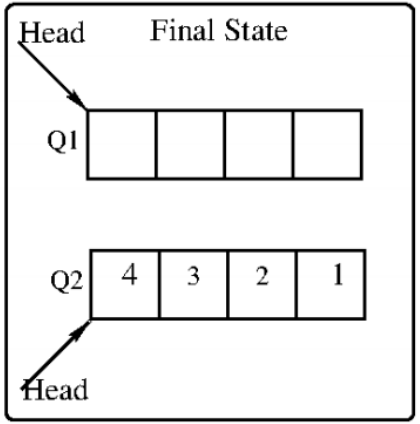


- s=pop()=62
- q=dequeue()=24
- s+q=62+24=86
- Ans:86

GATE CS 2022 | Question: 52

Consider the queues $\mathbf{Q_1}$ containing four elements and $\mathbf{Q_2}$ containing none (shown as the **Initial State** in the figure). The only operations allowed on these two queues are **Enqueue (Q, element)** and **Dequeue (Q)**. The minimum number of Enqueue operations on $\mathbf{Q_1}$ required to place the elements of $\mathbf{Q_1}$ in $\mathbf{Q_2}$ in reverse order (shown as the **Final State** in the figure) without using any additional storage is_____.





- Step 1:Dequeue (Q_1) , Enqueue (Q_2, 1)
- Step 2:Dequeue (Q_1) , Enqueue (Q_2, 2)
- Step 3:Dequeue (Q_2) , Enqueue (Q_2, 1)
- Step 4:Dequeue (Q_1) , Enqueue (Q_2, 3)
- Step 5:Dequeue (Q_2) , Enqueue (Q_2, 2)
- Step 6:Dequeue (Q_2) , Enqueue (Q_2, 1)
- Step 7:Dequeue (Q_1) , Enqueue (Q_2, 4)
- Step 8:Dequeue (Q_2) , Enqueue (Q_2, 3)
- Step 9:Dequeue (Q_2) , Enqueue (Q_2, 2)
- Step 10:Dequeue (Q_2) , Enqueue (Q_2, 1)
- Ans : 0

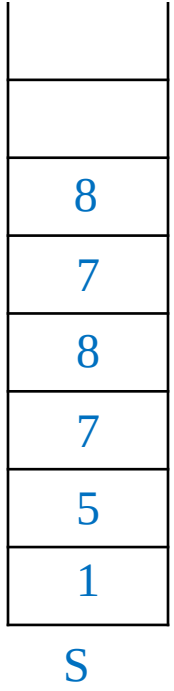
GATE CS 2023 | Question: 49

Consider a sequence a of elements a_0= 1, a_1=5, a_2= 7, a_3= 8, a_4= 9, and a_5= 2. The following operations are performed on a stack S and a queue Q, both of which are initially empty.

I. **push** the elements of **a** from $\mathbf{a_0}$ to $\mathbf{a_5}$ in that order into S.
II. **enqueue** the elements of **a** from $\mathbf{a_0}$ to $\mathbf{a_5}$ in that order into Q.
III. **pop** an element from S.
IV. **dequeue** an element from Q.
V. **pop** an element from S.
VI. **dequeue** an element from Q.
VII. **dequeue** an element from Q and push the same element into S.
VIII. **Repeat** operation VII three times.
IX. **pop** an element from S.
X. **pop** an element from S.

II IV VI VII

I
III
V
VIII
IX
X



The top element of S after executing the above operations is 8 .

Ans : 8

- **GATE DA 2024 | Question: 22**
- The fundamental operations in a double-ended queue D are:
- insertFirst(e) – Insert a new element e at the beginning of D.
- insertLast(e) – Insert a new element e at the end of D.
- removeFirst() – Remove and return the first element of D.
- removeLast() – Remove and return the last element of D.
- In an empty double-ended queue, the following operations are performed:
- insertFirst(10)
- insertLast(32)
- **a** ←removeFirst()
- insertLast(28)
- insertLast(17)
- **a** ←removeFirst()
- **a** ← removeLast()
- The value of **a** is _____.

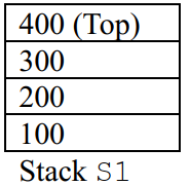
- a=10
- a=32
- a=17
- Ans : 17

GATE CS 2024 | Set 1 | Question: 23

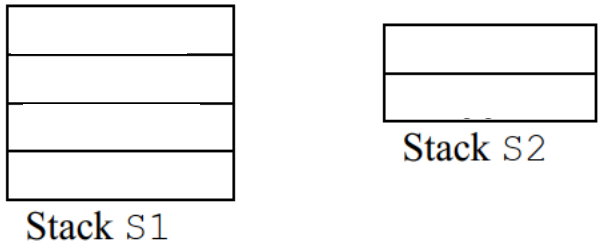
Operator	Precedence	Associativity
+	Highest	Left
-	High	Right
*	Medium	Right
/	Low	Right

- Consider the operator precedence and associativity rules for the *integer* arithmetic operators given in the table below.
- The value of the expression 3+1+5∗2/7+2−4−7−6/2 as per the above rules is _____.
- 3 1 + 5 + 2 * 7 2 + 4 7 6 - - - 2 / /
- (3+1)+5∗2/7+2−4−7−6/2
- =(4+5)∗2/7+2−4−7−6/2
- =9*2/(7+2)−4−7−6/2
- = 9*2/9−4−(7−6)/2
- = 9*2/9−(4−1)/2
- = 9*2/(9−3)/2
- = (9*2)/6/2
- =18/(6/2)
- =18/3
- =6
- Ans :6

- **GATE CS 2024 | Set 2 | Question: 38**
- Let S1 and S2 be two stacks. S1 has capacity of 4 elements. S2 has capacity of 2 elements. S1 already has 4 elements: 100, 200, 300, and 400, whereas S2 is empty, as shown below.
- Only the following three operations are available:
- PushToS2: Pop the top element from S1 and push it on S2.
- PushToS1: Pop the top element from S2 and push it on S1.
- GenerateOutput: Pop the top element from S1 and output it to the user.
- Note that the pop operation is not allowed on an empty stack and the push operation is not allowed on a full stack.Which of the following output sequences can be generated by using the above operations?

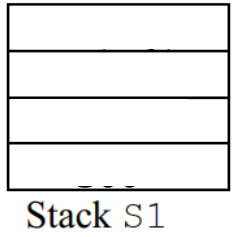


- (A) 100, 200, 400, 300 (B) 200, 300, 400, 100
- (C) 400, 200, 100, 300 (D) 300, 200, 400, 100

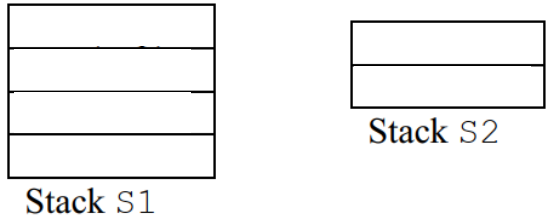


- (A) 100, 200, 400, 300
- Before 100 first you need to pop 200 so not possible
- (B) 200, 300, 400, 100
- PushToS2 , PushToS2 , GenerateOutput 200, PushToS1 , GenerateOutput 300 , PushToS1 , GenerateOutput 400 , GenerateOutput 100 , Possible

- (C) 400, 200, 100, 300
- GenerateOutput 400 , PushToS2, GenerateOutput 200 ,
- GenerateOutput 100, PushToS1, GenerateOutput 300
- Possible



- (D) 300, 200, 400, 100
- PushToS2 , GenerateOutput 300, GenerateOutput 200
- PushToS1, GenerateOutput 400, GenerateOutput 100
- Possible



- Ans:
- (B) 200, 300, 400, 100
- (C) 400, 200, 100, 300
- (D) 300, 200, 400, 100