DS assignment – 2

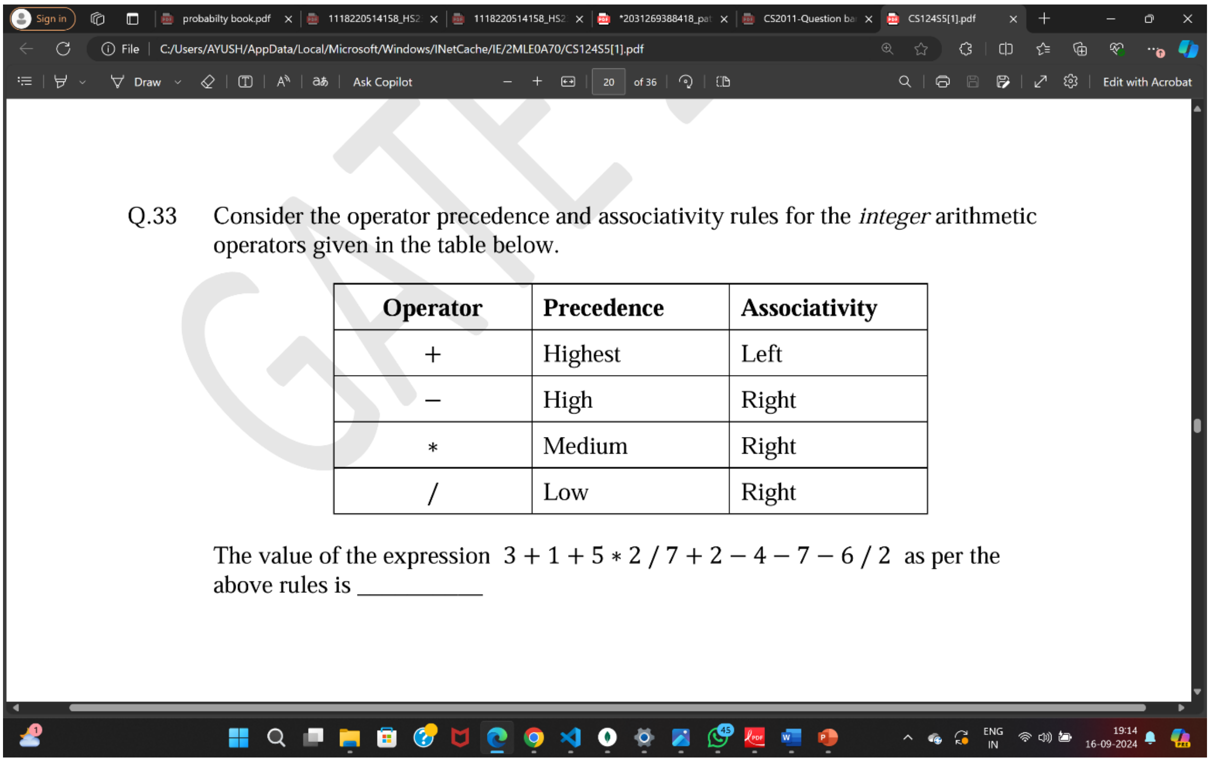
NAME : AYUSH PANDA

ROLL : 123CS0202

DATE : 16 \ 9 \ 2024

GATE – 2024

33)



Q.33 Consider the operator precedence and associativity rules for the *integer* arithmetic operators given in the table below.

Operator	Precedence	Associativity
+	Highest	Left
−	High	Right
∗	Medium	Right
/	Low	Right

The value of the expression $3 + 1 + 5 * 2 / 7 + 2 - 4 - 7 - 6 / 2$ as per the above rules is ________

INFIX TO POSTFIX -:

3 [] -> 3 [+] -> 3 1 [+] -> 3 1 + [+] -> 3 1 + 5 [+ *] -> 3 1 + 5 2 [+ *] -> 3 1 + 5 2* [+ /] ->

-> 3 1 + 5 2*7/2 [+] -> 3 1 + 5 2*7/2 +4 [+ -] -> ……… -> 3 1 + 5 2 * 7 / 2 + 4 - 7 - 6 2 / -

EVALUATING USING EMPTY STACK

[] -> [3] -> [3 , 1] -> (3 + 1 = 4) -> [4] -> [4 , 5] -> [4 , 5 , 2] -> [4 , 10] -> [4 , 10 , 7] -> (10 / 7)

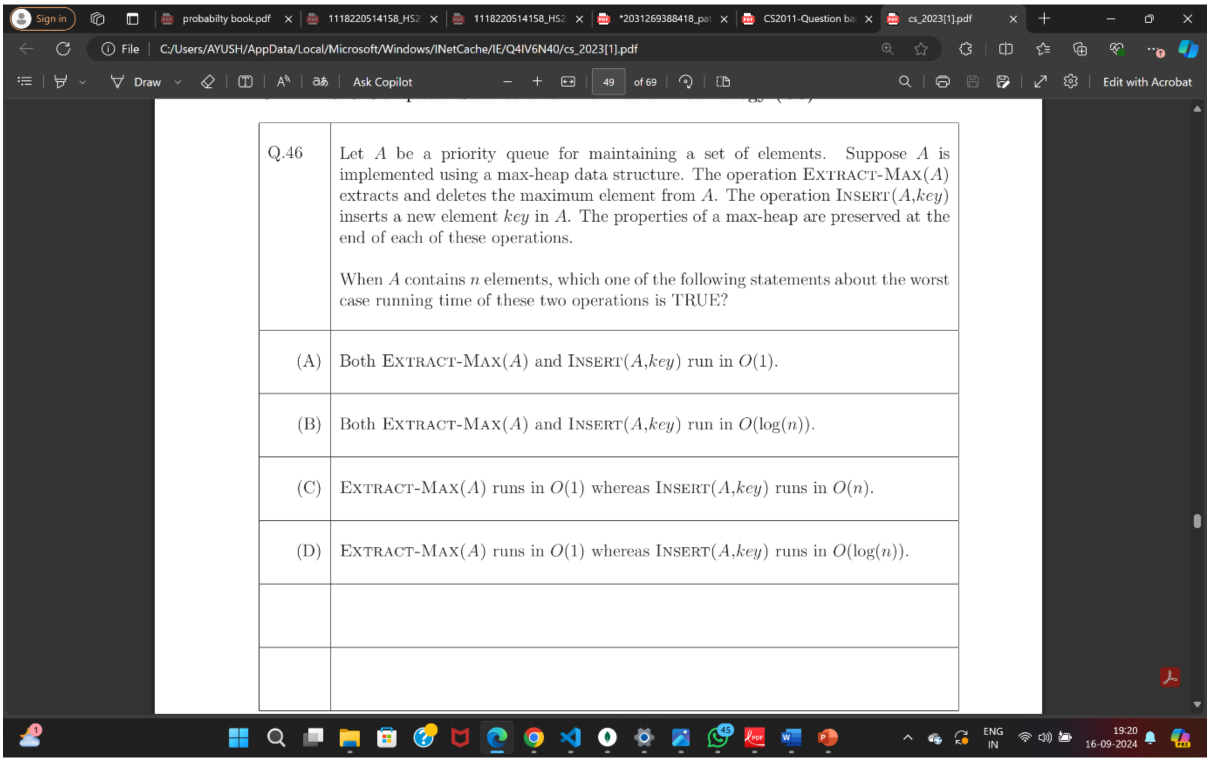
->[4 , 1.428] -> [4 , 1.428 , 2] -> [4 , 3.428] -> [4 , 3.428 , 4] -> (4 – 3.428)->[4 , 0.572] ->

->[4 , 0.572 , 7] -> [4 , -6.428] -> [4 , -6.428 , 6 , 2] -> [4 , -6.428 , 3] -> [4 , -9.428]

The resulting element in stack is -9.428

GATE-2023

46)



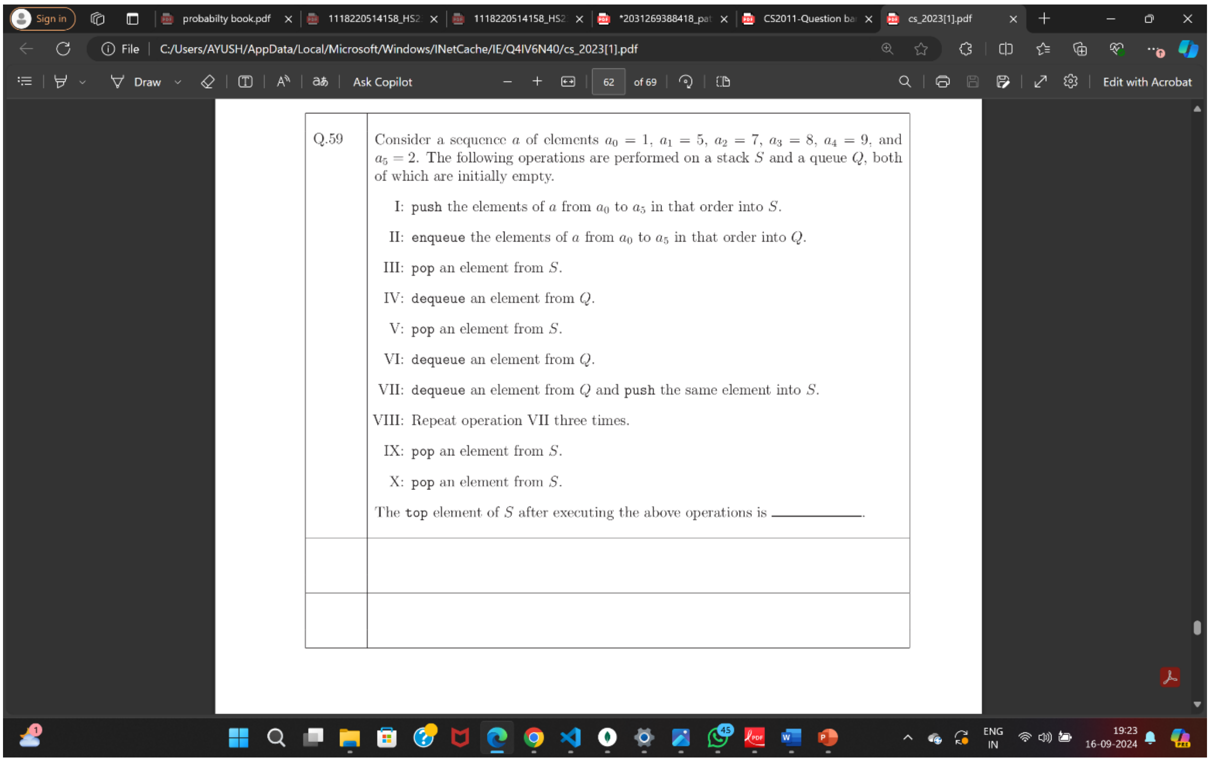
Q.46	Let A be a priority queue for maintaining a set of elements. Suppose A is implemented using a max-heap data structure. The operation EXTRACT-MAX(A) extracts and deletes the maximum element from A. The operation INSERT(A,key) inserts a new element key in A. The properties of a max-heap are preserved at the end of each of these operations. When A contains n elements, which one of the following statements about the worst case running time of these two operations is TRUE?
(A)	Both EXTRACT-MAX(A) and INSERT(A,key) run in $O(1)$.
(B)	Both EXTRACT-MAX(A) and INSERT(A,key) run in $O(\log(n))$.
(C)	EXTRACT-MAX(A) runs in $O(1)$ whereas INSERT(A,key) runs in $O(n)$.
(D)	EXTRACT-MAX(A) runs in $O(1)$ whereas INSERT(A,key) runs in $O(\log(n))$.

INSERT(A , key) -:

> When inserting a new element into a max-heap, it is initially added at the end of the heap (the bottom of the tree).
>
> Then, it is "heapified up" (swapped with its parent if it's larger) to restore the heap property (i.e., the parent node is always greater than or equal to its children).
>
> In the worst case, this heapify process will traverse up to the root of the heap, which is at most logn\log nlogn levels (since a heap is a complete binary tree). Hence, the time complexity is O(logn)O(\log n)O(logn).

EXTRACT-MAX(A) max element is at top so its simply removed ie O(1)

59)



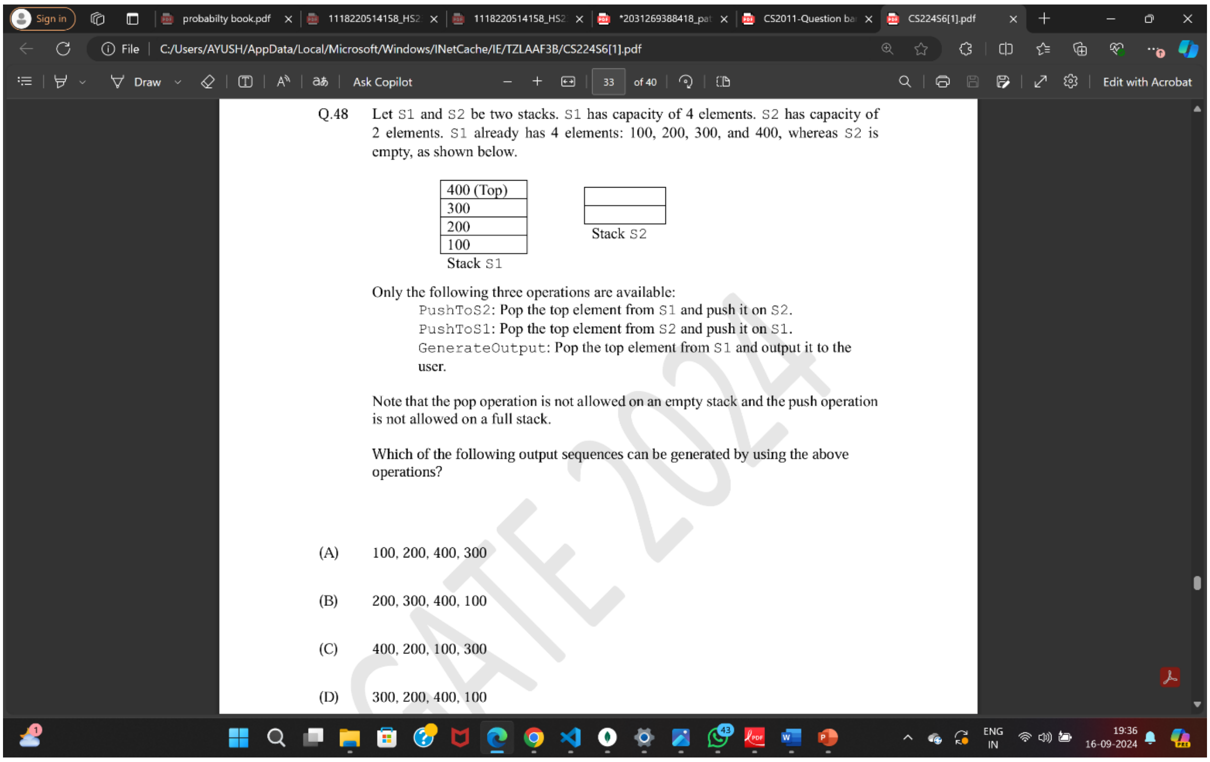
Q.59	Consider a sequence a of elements $a_0 = 1$, $a_1 = 5$, $a_2 = 7$, $a_3 = 8$, $a_4 = 9$, and $a_5 = 2$. The following operations are performed on a stack S and a queue Q, both of which are initially empty. I: push the elements of a from a_0 to a_5 in that order into S. II: enqueue the elements of a from a_0 to 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. The top element of S after executing the above operations is ______.

State of stack and queue are as follows -:

1) Empty Stack () , Empty Queue ()
2) S (1 , 5 , 7 , 8 , 9 , 2) , Q ()
3) S (1 , 5 , 7 , 8 , 9 , 2) , Q (1 , 5 , 7 , 8 , 9 , 2)

4) S (1 , 5 , 7 , 8 , 9) , Q (5 , 7 , 8 , 9 , 2)
5) S (1 , 5 , 7 , 8) , Q (7 , 8 , 9 , 2)
6) S (1 , 5 , 7 , 8 , 7) , Q (8 , 9 , 2)
7) S (1 , 5 , 7 , 8 , 7 , 8 , 9 , 2) , Q ( )
8) S (1 , 5 , 7 , 8 , 7 , 8) , Q ( )
9) Clearly top element is 8

GATE-2024-cs2-62)



Q.48 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.

400 (Top)
300
200
100

Stack S1

Stack S2

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

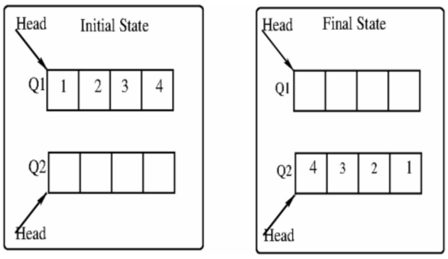
Option – C

Our operations are as follows -:

1) GenerateOutput
2) PushToS2
3) GenerateOutput
4) GenerateOutput
5) PushToS1
6) GenerateOutput

62)

Q.62 Consider the queues Q_1 containing four elements and 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 Q_1 required to place the elements of Q_1 in Q_2 in reverse order (shown as the Final State in the figure) without using any additional storage is__________.



Q1 = [1 , 2 , 3 , 4] , Q2 = [] -> e = deq(Q1) enq(Q2,e) ->

Q1 = [2 , 3 , 4] , Q2 = [1] -> e = deq(Q1) enq(Q2,e) ->

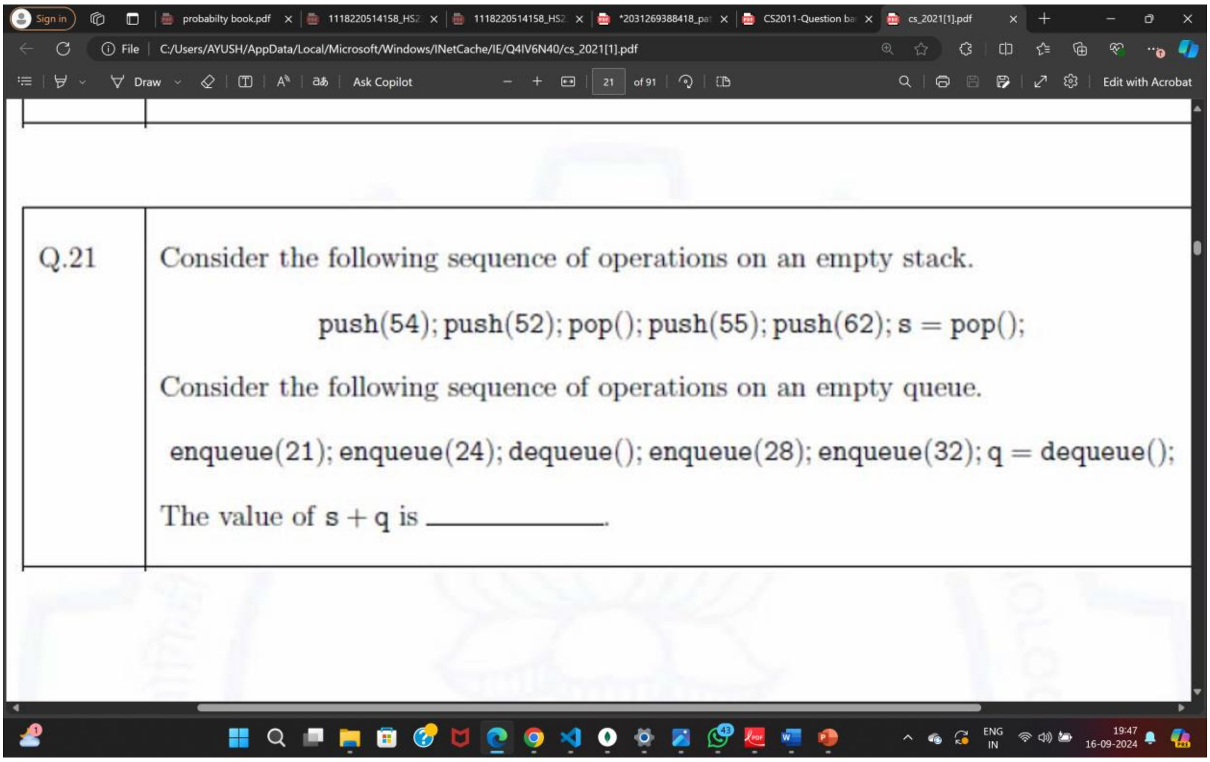
Q1 = [3 , 4] , Q2 = [2 , 1] -> e = deq(Q1) enq(Q2,e) ->

Q1 = [4] , Q2 = [3 , 2 , 1] -> e = deq(Q1) enq(Q2,e) ->

Q1 = [ ] , Q2 = [4 , 3 , 2 , 1]

GATE – 2021

21)



Q.21 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 ________.

Empty stack () -> (54) -> (54 , 52) -> (54) -> (54, 55) -> (54 , 55 , 52) -> (54 , 55)

S = 52

Empty queue () -> (21) -> (21 , 24) -> (24) -> (24 , 28) -> (24 , 28 , 32) -> (28 , 32)

Q = 24

S + q = 76 ;