# CSE/IT Batch-Hinglish

# Data Structure

## Stacks and Queues

**DPP-01**

**[NAT]**

**1.** Consider the following sequence of operations on an empty stack:

push (5); push (2); pop(); push(4); push(6); p=pop(); q=pop(); r=pop();

The value of p+q-r is-_____.

**[MCQ]**

**2.** Which of the following includes the applications of stack?

(a) Recursive function calls
(b) HTML and XML Tag matching
(c) Checking if an expression contains balanced parantheses.
(d) Finding the maximum element in a given sequence.

**[NAT]**

**3.** A stack is implemented using array. S represents the pointer to the top element in the stack. Initially the stack contains the elements: a(top), b. Assume Push(S, i) push an element i into the stack at index S. Whenever a Push operation will be performed, it will returns S++ after the push operation. Pop() pops the topmost element and returns the next top index. Top() is a function that returns the topmost element of the stack. Consider the following statements:

P: Top(Pop((Pop(Pop((Push(Push(S, c), d)))))))= a

Q: Pop(Pop(Pop(Pop(Push(Pop(Push(S, c)), d))))) = a

Which of the following statements is/are INVALID?

(a) P only (b) Q only
(c) Both P and Q (d) Neither P nor Q

**[MCQ]**

**4.** A single array A[1…MAXSIZE] is used to implement two stacks. The two stacks grow from opposite ends of the array. Variables top1 and top2 (topl< top 2) point to the location of the topmost element in each of the stacks. If the space is to be used efficiently, the condition for "stack full" is-

(a) (top1=MAXSIZE/2)and(top2=MAXSIZE/2+1)
(b) (top1=MAXSIZE/2)or(top2=MAXSIZE/2+1)
(c) top1 + top2 = MAXSIZE
(d) top1=top2-1

**[NAT]**

**5.** A stack is implemented using a singly linked list that uses node structure-

```
struct node
{
 int data;
struct node *next;
}
node;
```

Let *head denote the address of the start node respectively. Assume, the stack is not empty. Consider the following function that intends to delete the topmost element of the stack:

```
node * f(node *head)
{
      node *p=head; _____;
      free(p);
      p=NULL;
   }
```

The missing blank is-

(a) while(p → next!=NULL) p=p→next;
(b) p=p → next;
(c) head=head → next;
(d) None

**[MSQ]**

**6.** Which one of the following permutations cannot be obtained in the output string using a stack and assuming that the input sequence is a, b, c, d, e in the same order?

(a) c d e a b (b) a e b c d
(c) c d e b a (d) e d c b a

**[MCQ]**

**7.** A stack is implemented using array of size 4. S represents the pointer to the top element in the stack. Initially the stack contains the elements- a(top), b. Assume Push(S, i) push an element i into the stack at index S. Whenever a Push operation will be performed, it will returns S++ after the push operation. Pop() pops the topmost element and returns the next top index. isEmpty() returns TRUE if the stack is empty. isFull() returns TRUE if the stack is full. Consider the following statements:

P: isFull(Push(Pop(Push(Push(S, c), d))), e))= TRUE

Q: isEmpty(Push(Pop(Pop(Push(Pop(Push(S, c)), d))))), e) = FALSE

Which of the following statements is/are VALID?

(a) P only (b) Q only
(c) Both P and Q (d) Neither P nor Q

**[NAT]**

**8.** Let S be a stack of size n≥1. Starting with the empty stack, suppose we push the first *5* natural numbers in sequence, and then perform 5 pop operations. Assume that Push and Pop operations take 3 seconds each, and 1 seconds elapse between the end of one such stack operation and the start of the next operation. The average stack-life of an element of this stack is _____.

# Answer Key

1. **(5)**
2. **(a, b, c)**
3. **(c)**
4. **(d)**
5. **(c)**
6. **(a, b)**
7. **(c)**
8. **(17)**

# Hints and Solutions

**1. (5)**

push (5);
push (2);
pop(); //2 is popped
push(4);
push(6);
p=pop(); //6 is popped
q=pop(); //4 is popped
r=pop(); //5 is popped

The final value of p+q-r = 6+4-5= 5

**2. (a, b, c)**

The application of stack:
Recursive Function Calls
HTML and XML Tag matching
Checking if an expression contains balanced parantheses.

**3. (c)**

Stack already contains a(top), b.
top(pop((pop(pop((push(push(S, c), d)))))))
It pushes c into the stack. It pushes d into the stack.
It pops d. It pops c.
It pops a.
Top contains b now.
pop(pop(pop(pop(push(pop(push(S, c)), d))))) = a
It pushes c and pops it.
It pushes d and pops it.
It pops a and then b.
Last pop operation cannot be implemented as the stack is already empty.

**4. (d)**

If the stacks are growing from two ends, top2-top1=1.

**5. (c)**

If a stack is implemented using linked list, push and pop occurs from one default location-head/start of the linked list.

The missing statement include-
head=head->next

**6. (a, b)**

(a) is NOT possible
Push a;
Push b;
Push c;
Pop c;
Push d;
Pop d;
Push e;
Pop e;
Pop a; (Not possible) top contains b.

(b) is NOT possible.
Push a;
Pop a;
Push b;
Push c;
Push d;
Push e;
Pop e;
Pop b;(Not possible) top contains d.

**7. (c)**

isFull(push(pop(push(push(S, c), d))), e))= TRUE

Push c;
Push d;
Pop;
Push e;
The stack is full (b, a, c, e(top)).
is Empty(push(pop(pop(push(pop(push(S, c)), d))))), e) = FALSE
Push c;
Pop c;
Push d;
Pop d;
Pop a;
Push e;(Stack contains b, e(top))
The stack is not empty.

**8. (17)**

Stack Life time of 5=1
Stack Life time of =2*3+3*1=9
Stack Life time of 3=4*3+5*1=17
Stack Life time of 2=6*3+7*1=25
Stack Life time of 1=8*3+9*1=33
Average stack-life of an element =
(1+9+17+25+33)/5 = 17