SANFOUNDRY
Global Education & Learning

Data Structure Questions and Answers – Stack using Linked List

« Prev Next »

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Stack using Linked List”.

1. What is the best case time complexity of deleting a node in Singly Linked list?
a) O (n)
b) O (n^2)
c) O (nlogn)
d) O (1)
View Answer

Answer: d
Explanation: Deletion of the head node in the linked list is taken as the best case. The successor of the head node is changed to head and deletes the predecessor of the newly assigned head node. This process completes in O(1) time.

advertisement

2. Which of the following statements are not correct with respect to Singly Linked List(SLL) and Doubly Linked List(DLL)?
a) Complexity of Insertion and Deletion at known position is O(n) in SLL and O(1) in DLL
b) SLL uses lesser memory per node than DLL
c) DLL has more searching power than SLL
d) Number of node fields in SLL is more than DLL
View Answer

Answer: d
Explanation: To insert and delete at known positions requires complete traversal of the list in worst case in SLL, SLL consists of an item and a node field, while DLL has an item and two node fields, hence SLL occupies lesser memory, DLL can be traversed both ways(left and right), while SLL can traverse in only one direction, hence more searching power of DLL. Node fields in SLL is 2 (data and address of next node) whereas in DLL is 3(data, address to next node, address to previous node).

3. Given below is the Node class to perform basic list operations and a Stack class with a no arg constructor.

Select from the options the appropriate pop() operation that can be included in the Stack class. Also ‘first’ is the top-of-the-stack.

```
class Node
{
        protected Node next;
        protected Object ele;
        Node()
        {
                this(null,null);
        }
        Node(Object e,Node n)
        {
                ele=e;
                next=n;
        }
        public void setNext(Node n)
        {
                next=n;
        }
        public void setEle(Object e)
        {
                ele=e;
        }
        public Node getNext()
        {
                return next;
        }
        public Object getEle()
        {
                return ele;
        }
}

class Stack
{
        Node first;
        int size=0;
        Stack()
        {
                first=null;
        }
}
```

a)

```
public Object pop()
{
        if(size == 0)
        System.out.println("underflow");
        else
        {
                Object o = first.getEle();
                first = first.getNext();
                size--;
                return o;
        }
}
```

b)

```
public Object pop()
{
        if(size == 0)
        System.out.println("underflow");
        else
        {
                Object o = first.getEle();
                first = first.getNext().getNext();
                size--;
                return o;
        }
}
```

c)

advertisement

```
public Object pop()
{
        if(size == 0)
        System.out.println("underflow");
        else
        {
                first = first.getNext();
                Object o = first.getEle();
                size--;
                return o;
        }
}
```

d)

```
public Object pop()
{
        if(size == 0)
        System.out.println("underflow");
        else
        {
                first = first.getNext().getNext();
                Object o = first.getEle();
                size--;
                return o;
        }
}
```

View Answer

Answer: a

Explanation: pop() should return the Object pointed to by the node ‘first’. The sequence of operations is, first, get the element stored at node ‘first’ using getEle(), and second, make the node point to the next node using getNext().

4. What does the following function do?

```
public Object some_func()throws emptyStackException
{
        if(isEmpty())
                throw new emptyStackException("underflow");
        return first.getEle();
}
```

a) pop
b) delete the top-of-the-stack element
c) retrieve the top-of-the-stack element
d) push operation

View Answer

Answer: c

Explanation: This code is only retrieving the top element, note that it is not equivalent to pop operation as you are not setting the ‘next’ pointer point to the next node in sequence.

5. What is the functionality of the following piece of code?

```
public void display()
{
        if(size == 0)
                System.out.println("underflow");
        else
        {
                Node current = first;
                while(current != null)
                {
                        System.out.println(current.getEle());
                        current = current.getNext();
                }
        }
}
```

a) reverse the list
b) display the list
c) display the list excluding top-of-the-stack-element
d) reverse the list excluding top-of-the-stack-element

View Answer

Answer: b
Explanation: An alias of the node 'first' is created which traverses through the list and displays the elements.

advertisement

6. What does 'stack overflow' refer to?
a) accessing item from an undefined stack
b) adding items to a full stack
c) removing items from an empty stack
d) index out of bounds exception

View Answer

Answer: b
Explanation: Adding items to a full stack is termed as stack underflow.

7. Given below is the Node class to perform basic list operations and a Stack class with a no arg constructor. Select from the options the appropriate push() operation that can be included in the Stack class. Also 'first' is the top-of-the-stack.

```
class Node
{
        protected Node next;
        protected Object ele;
        Node()
        {
                this(null,null);
        }
        Node(Object e,Node n)
        {
                ele=e;
                next=n;
        }
        public void setNext(Node n)
        {
                next=n;
        }
        public void setEle(Object e)
        {
                ele=e;
        }
        public Node getNext()
        {
                return next;
        }
        public Object getEle()
        {
                return ele;
        }
}

class Stack
{
        Node first;
        int size=0;
        Stack()
        {
                first=null;
        }
}
```

a)

```
public void push(Object item)
{
        Node temp = new Node(item,first);
        first = temp;
        size++;
}
```

b)

```
public void push(Object item)
{
        Node temp = new Node(item,first);
        first = temp.getNext();
        size++;
}
```

c)

```
public void push(Object item)
{
        Node temp = new Node();
        first = temp.getNext();
        first.setItem(item);
        size++;
}
```

d)

```
public void push(Object item)
{
        Node temp = new Node();
        first = temp.getNext.getNext();
        first.setItem(item);
        size++;
}
```

View Answer

Answer: a
Explanation: To push an element into the stack, first create a new node with the next pointer point to the current top-of-the-stack node, then make this node as top-of-the-stack by assigning it to 'first'.

8. Consider these functions:
push() : push an element into the stack
pop() : pop the top-of-the-stack element
top() : returns the item stored in top-of-the-stack-node
What will be the output after performing these sequence of operations

```
push(20);
push(4);
top();
pop();
pop();
pop();
push(5);
top();
```

a) 20
b) 4
c) stack underflow
d) 5

View Answer

Answer: d
Explanation: 20 and 4 which were pushed are popped by the two pop() statements, the recent push() is 5, hence top() returns 5.

9. Which of the following data structures can be used for parentheses matching?
a) n-ary tree
b) queue
c) priority queue
d) stack

View Answer

Answer: d
Explanation: For every opening brace, push it into the stack, and for every closing brace, pop it off the stack. Do not take action for any other character. In the end, if the stack is empty, then the input has balanced parentheses.

10. Minimum number of queues to implement stack is ___________
a) 3
b) 4
c) 1
d) 2
View Answer

Answer: c
Explanation: Use one queue and one counter to count the number of elements in the queue.

Sanfoundry Global Education & Learning Series – Data Structure.

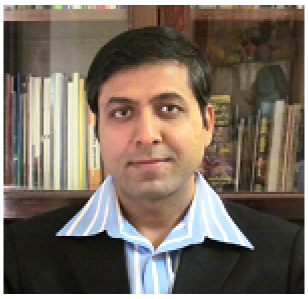
To practice all areas of Data Structure, here is complete set of 1000+ Multiple Choice Questions and Answers.

« Prev - Data Structure Questions and Answers – Stack using Array

» Next - Data Structure Questions and Answers – Queue using Array

Recommended Posts:

11. C Programming Examples without using Recursion
12. Java Programming Examples on Collections
13. C Programming Examples on Stacks & Queues
14. Java Programming Examples on Data-Structures
15. C Programming Examples on Data-Structures
16. C++ Programming Examples on Data-Structures
17. Data Structure Questions and Answers
18. Python Programming Examples on Linked Lists
19. C# Programming Examples on Data Structures
20. C Programming Examples on Linked List



Manish Bhojasia, a technology veteran with 20+ years @ Cisco & Wipro, is Founder and CTO at Sanfoundry. He is Linux Kernel Developer & SAN Architect and is passionate about competency developments in these areas. He lives in Bangalore and delivers focused training sessions to IT professionals in Linux Kernel, Linux Debugging, Linux Device Drivers, Linux Networking, Linux Storage, Advanced C Programming, SAN Storage Technologies, SCSI Internals & Storage Protocols such as iSCSI & Fiber Channel. Stay connected with him @ LinkedIn