About Submissions

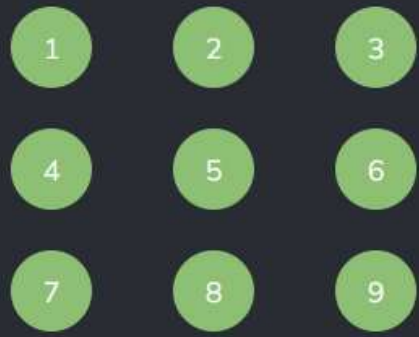
Q1. Delete In A Linked List

Consider a singly linked list of the form where F is a pointer to the first element in the linked list and L is the pointer to the last element in the list. The time of which of the following operations depends on the length of the list?

- Delete the element from the last
- Delete the element from first
- Add an element to the last
- Interchange the first two elements

Previous Next

Result



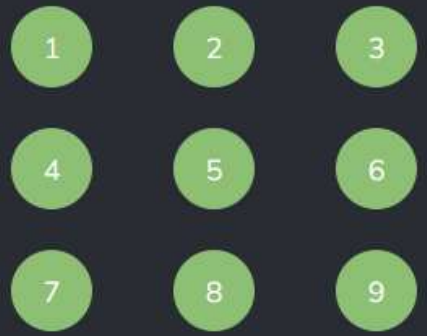
Q2. Kth element in a Linked List

In what best complexity you can find the kth element from last in a linked list?

- Time: O(n^2) Space: O(1)
- Time: O(n) Space: O(n)
- Time: O(n) Space: O(1)
- None

Previous Next

Result



Q3. Basics of Linked List

Which of the following points is/are true about Linked List data structure when it is compared with array

- Arrays have better cache locality that can make them better in terms of performance.
- It is easy to insert and delete elements in Linked List
- Random access is not allowed in a typical implementation of Linked Lists
- The size of array has to be pre-decided, linked lists can change their size any time
- All of above

Previous Next

Result

1 2 3
4 5 6
7 8 9

Q4. List concatenation

Result

The concatenation of two list can performed in O(1) time. Which of the following variation of linked list can be used?

1 2 3

4 5 6

7 8 9

- Singly linked list
- Doubly linked list
- Circular doubly linked list
- Array implementation of list

Previous Next

Q5. Sort a list

Which of the following sorting algorithms can be used to sort a random linked list with minimum time complexity?

- insertion sort
- merge sort
- quick sort
- heap sort

Previous Next

Result

1 2 3
4 5 6
7 8 9

Q6. Solve for Linked List

Predict the output that the following code gives for a linked list given as **1->2->3->4->5->6**

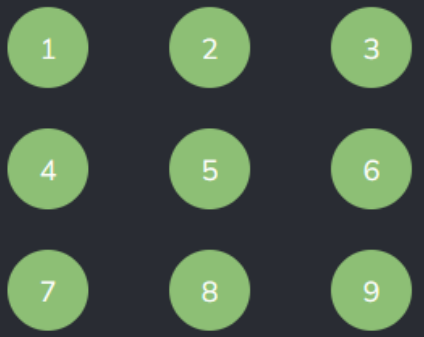
cpp

```
void fun(node* start)
{
    if(start == NULL)
    return;
    cout<<start->data<<" ";
    if(start->next != NULL )
    fun(start->next->next);
    cout<<start->data<<" ";
}
```

Java

```
void fun(node start)
{
    if(start == null)
    return;
    System.out.print(start.data+" ");
    if(start.next != null )
    fun(start.next.next);
    System.out.print(start.data+" ");
}
```

Result




146641
135135
1235
135531
Previous
Next

Q7. Skip List

Result

Skip lists are similar to which of the following data structure? Read about Skip list from here

- Stack
- Heap
- BST
- Balanced BST

1 2 3
4 5 6
7 8 9

Previous Next

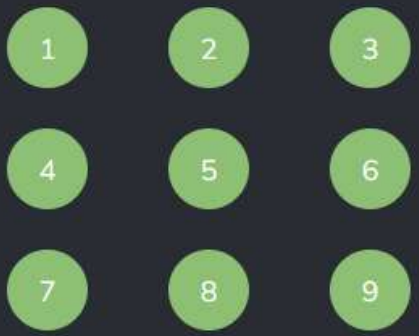
Q8. Memory Efficient List

What is a memory efficient double linked list?

- [x] Each node has only one pointer to traverse the list back and forth
- [ ] The list has breakpoints for faster traversal
- [ ] An auxiliary singly linked list acts as a helper list to traverse through the doubly linked list
- [ ] None

Previous Next

Result



Q9. Solve for Linked List 2

Predict the modified doubly linked list of the following code for the given doubly linked list as 1<--> 2 <--> 3 <--> 4 <--> 5 <-->6

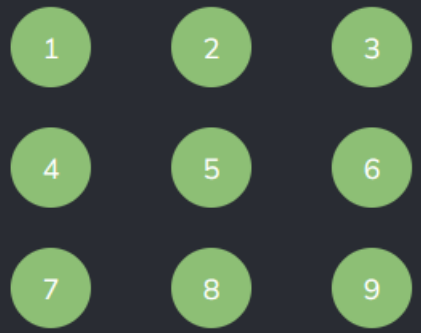
CPP

```
void fun(node **head_ref)
{
    node *temp = NULL;
    node *current = *head_ref;

    while (current !=  NULL)
    {
        temp = current->prev;
        current->prev = current->next;
        current->next = temp;
        current = current->prev;
    }

    if(temp != NULL )
        *head_ref = temp->prev;
}
```

JAVA

```
void fun(node head_ref)
{
    node temp = null;
    node current = head_ref;
```

Result



JAVA

```
void fun(node head_ref)
{
    node temp = null;
    node current = head_ref;

    while (current !=  null)
    {
        temp = current.prev;
        current.prev = current.next;
        current.next = temp;
        current = current.prev;
    }

    if(temp != null )
        head_ref = temp.prev;
}
```

- 2 <--> 1 <--> 4 <--> 3 <--> 6 <-->5
- 5 <--> 4 <--> 3 <--> 2 <--> 1 <-->6
- 6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1
- 6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2