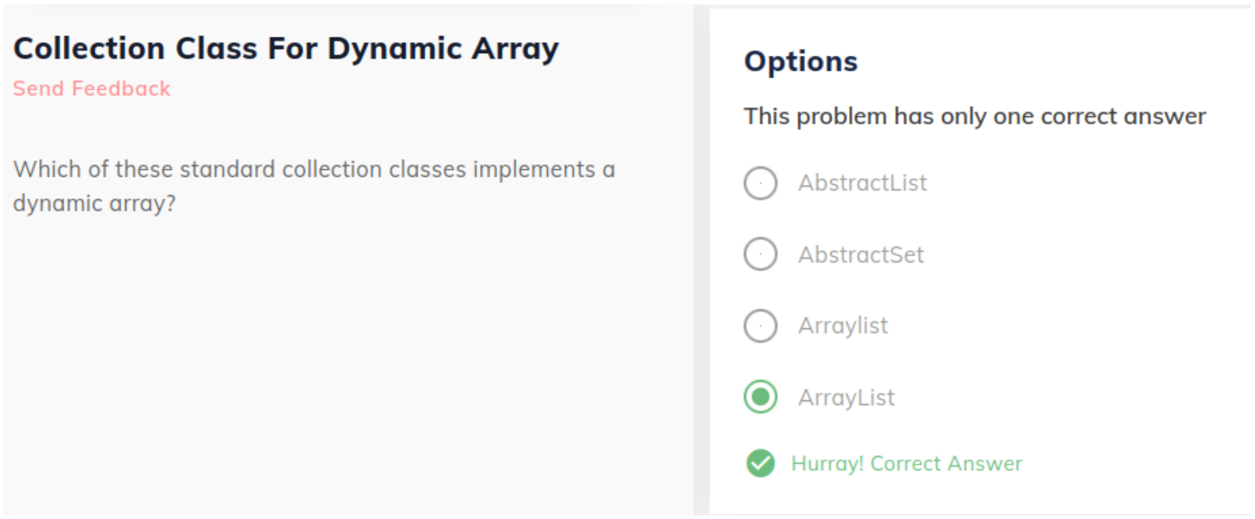
Linked List 1



Collection Class For Dynamic Array

Send Feedback

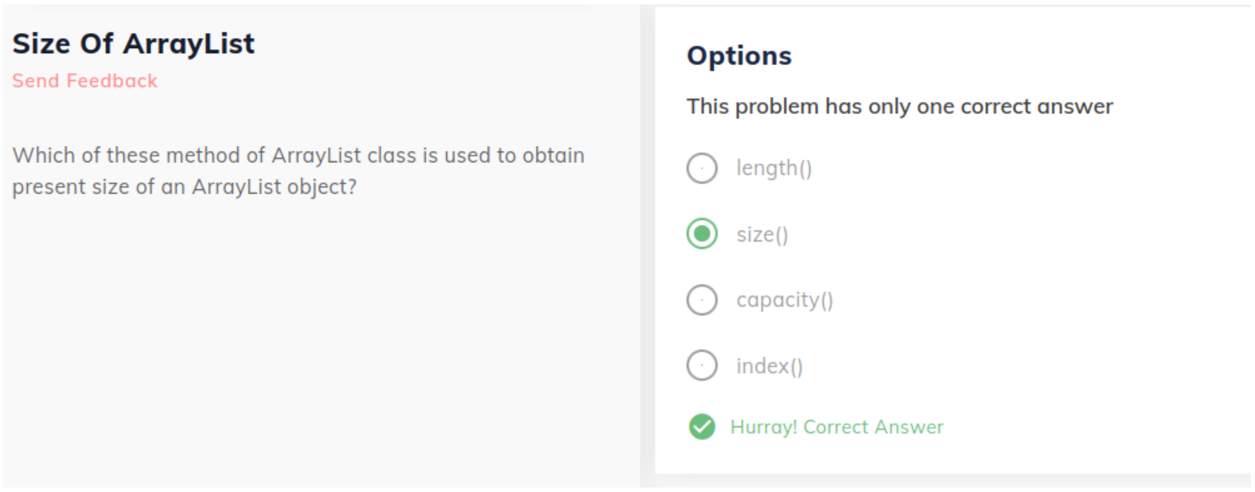
Which of these standard collection classes implements a dynamic array?

Options

This problem has only one correct answer

- AbstractList
- AbstractSet
- Arraylist
- [✓ ArrayList]

✓ Hurray! Correct Answer



Size Of ArrayList

Send Feedback

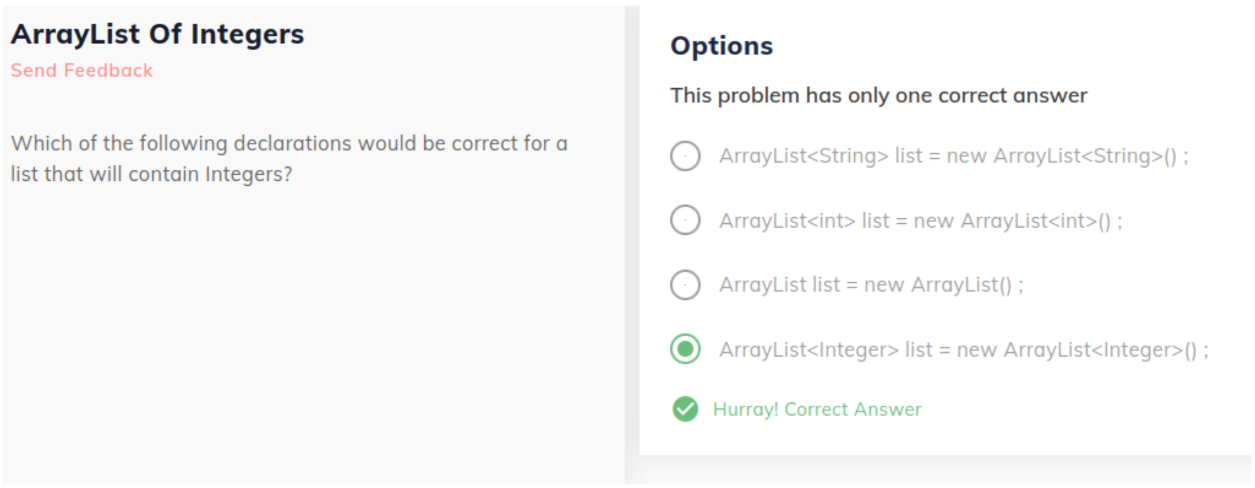
Which of these method of ArrayList class is used to obtain present size of an ArrayList object?

Options

This problem has only one correct answer

- length()
- [✓ size()]
- capacity()
- index()

✓ Hurray! Correct Answer



ArrayList Of Integers

Send Feedback

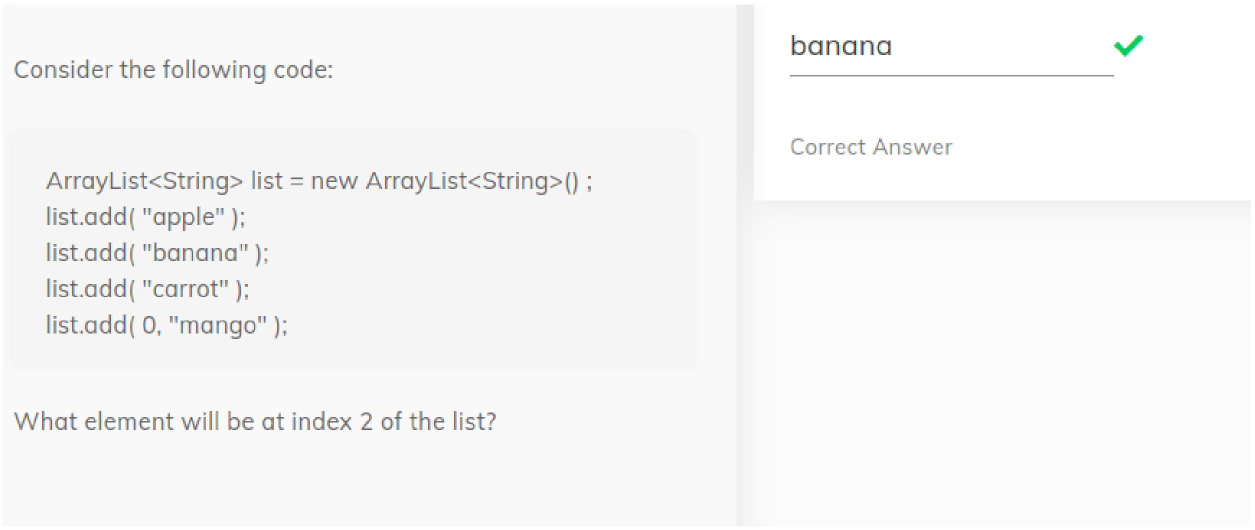
Which of the following declarations would be correct for a list that will contain Integers?

Options

This problem has only one correct answer

- ArrayList<String> list = new ArrayList<String>() ;
- ArrayList<int> list = new ArrayList<int>() ;
- ArrayList list = new ArrayList() ;
- [✓ ArrayList<Integer> list = new ArrayList<Integer>() ;]

✓ Hurray! Correct Answer



Consider the following code:

```
ArrayList<String> list = new ArrayList<String>() ;
list.add( "apple" );
list.add( "banana" );
list.add( "carrot" );
list.add( 0, "mango" );
```

What element will be at index 2 of the list?

banana ✓

Correct Answer

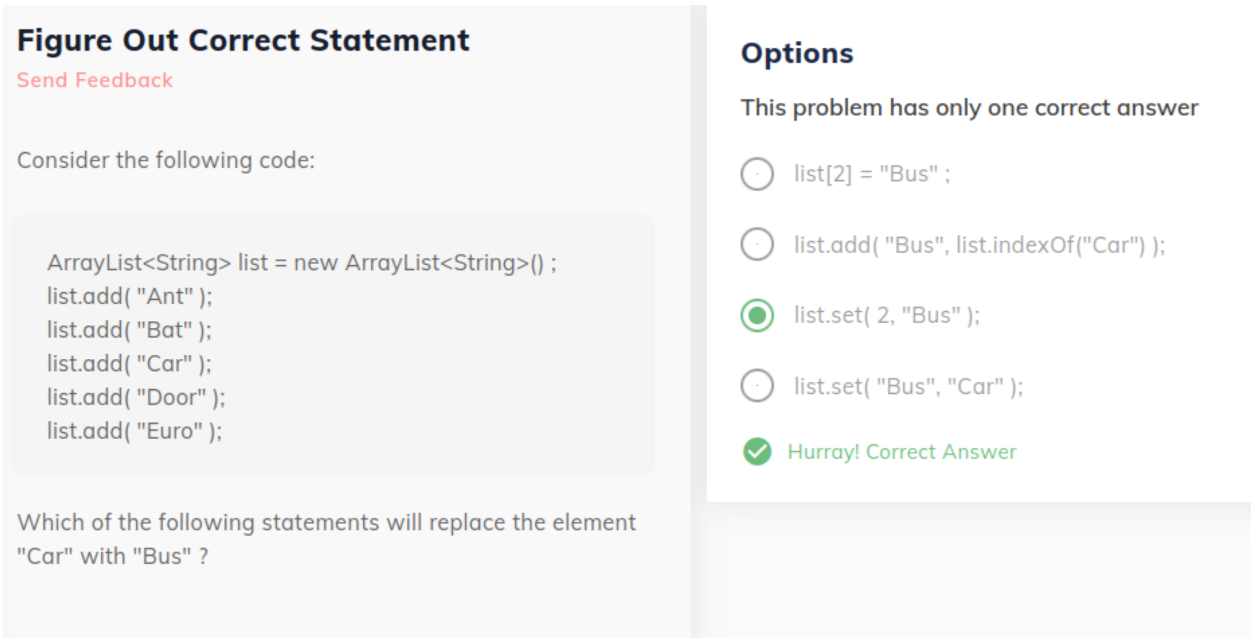


Figure Out Correct Statement

Send Feedback

Consider the following code:

```
ArrayList<String> list = new ArrayList<String>() ;
list.add( "Ant" );
list.add( "Bat" );
list.add( "Car" );
list.add( "Door" );
list.add( "Euro" );
```

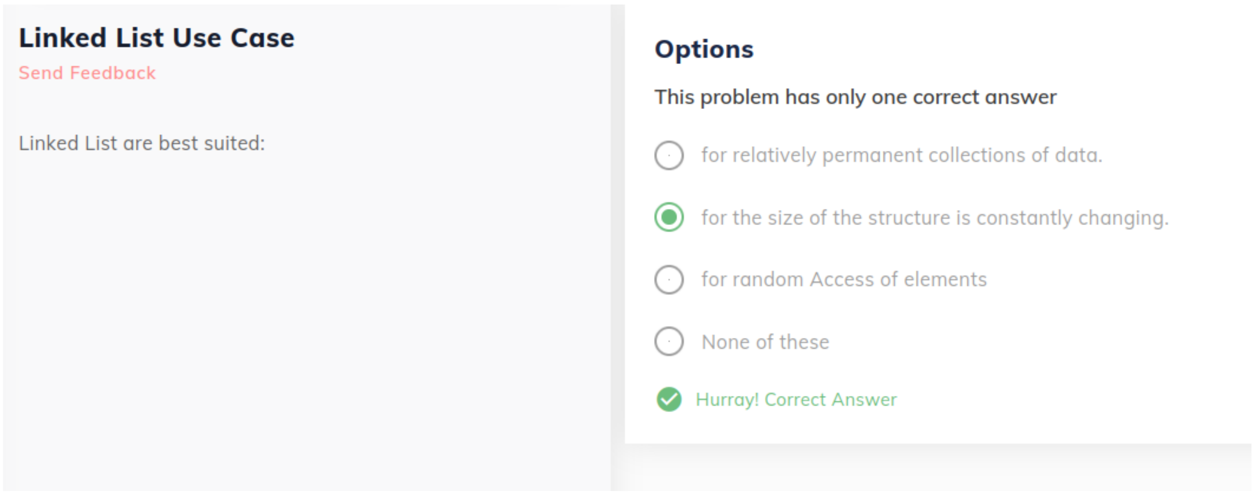
Which of the following statements will replace the element "Car" with "Bus" ?

Options

This problem has only one correct answer

- list[2] = "Bus" ;
- list.add("Bus", list.indexOf("Car"));
- [✓ list.set(2, "Bus");]
- list.set("Bus", "Car");

✓ Hurray! Correct Answer



Linked List Use Case

Send Feedback

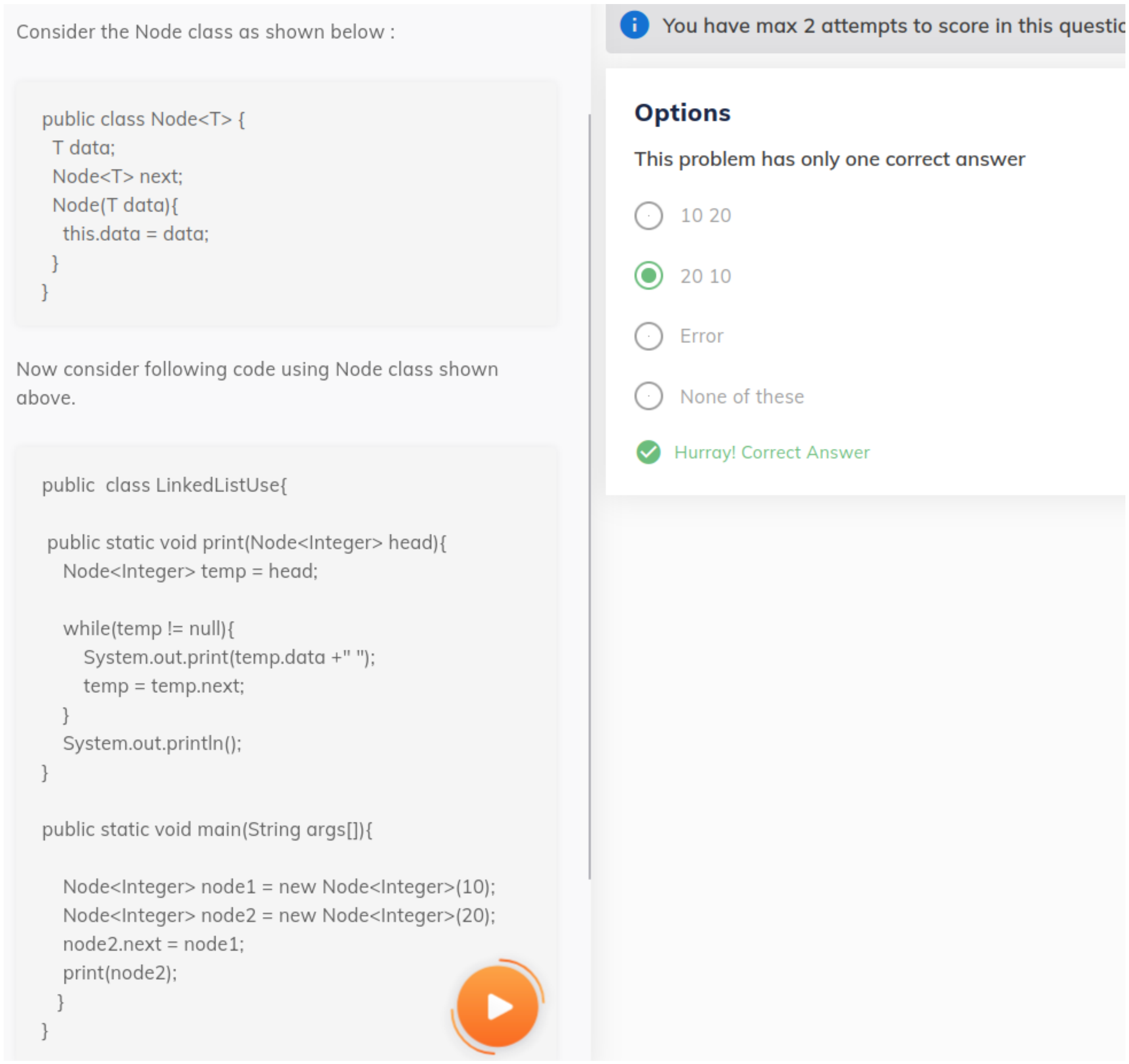
Linked List are best suited:

Options

This problem has only one correct answer

- for relatively permanent collections of data.
- [✓ for the size of the structure is constantly changing.]
- for random Access of elements
- None of these

✓ Hurray! Correct Answer


Consider the Node class as shown below :

```
public class Node<T> {
  T data;
  Node<T> next;
  Node(T data){
    this.data = data;
  }
}
```

Now consider following code using Node class shown above.

```
public  class LinkedListUse{

 public static void print(Node<Integer> head){
    Node<Integer> temp = head;

    while(temp != null){
       System.out.print(temp.data +" ");
       temp = temp.next;
    }
    System.out.println();
 }

 public static void main(String args[]){

    Node<Integer> node1 = new Node<Integer>(10);
    Node<Integer> node2 = new Node<Integer>(20);
    node2.next = node1;
    print(node2);
  }
}
```

You have max 2 attempts to score in this questic

Options

This problem has only only one correct answer

- 10 20
- [✓ 20 10]
- Error
- None of these

✓ Hurray! Correct Answer

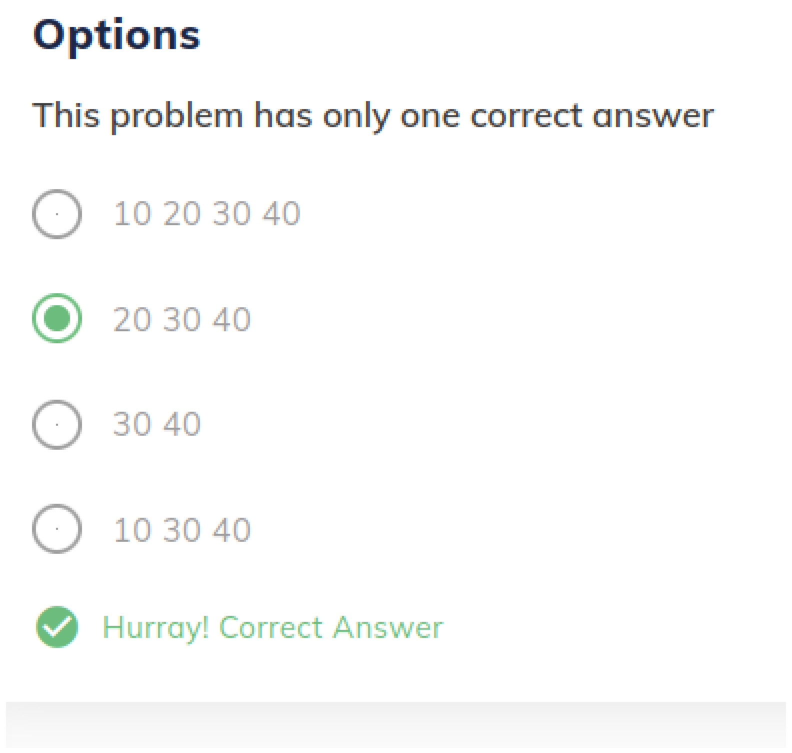
Consider the Node class as shown below :

```
public class Node<T> {
  T data;
  Node<T> next;
  Node(T data){
    this.data = data;
  }
}
```

Now consider following code using Node class shown above.

```
public  class LinkedListUse{

 public static void print(Node<Integer> head){
    Node<Integer> temp = head;
    while(temp != null){
        System.out.print(temp.data +" ");
        temp = temp.next;
    }
    System.out.println();
}

public static void main(String args[]){

    Node<Integer> node1 = new Node<Integer>(10);
    Node<Integer> node2 = new Node<Integer>(20);
    Node<Integer> node3 = new Node<Integer>(30);
    Node<Integer> node4 = new Node<Integer>(40);
    node1.next = node2;
    node2.next = node3;
    node3.next = node4;
    print(node2);
  }
}
```

What will be the Output ?



Options

This problem has only one correct answer

- 10 20 30 40
- [✓ 20 30 40]
- 30 40
- 10 30 40

Hurray! Correct Answer

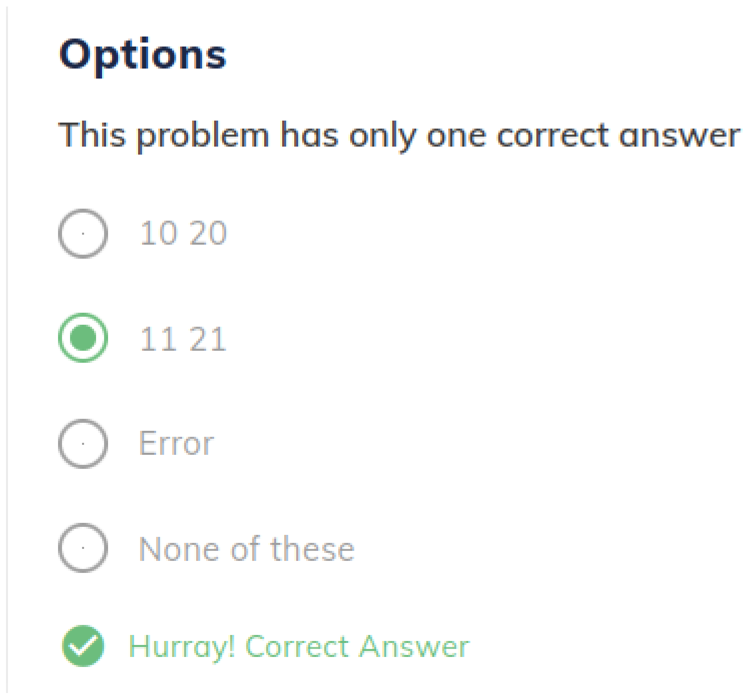
Consider the Node class as shown below :

```
public class Node<T> {
  T data;
  Node<T> next;
  Node(T data){
    this.data = data;
  }
}
```

Now consider following code using Node class shown above.

```
public  class LinkedListUse{

 public static void print(Node<Integer> head){
    Node<Integer> temp = head;

    while(temp != null){
        System.out.print(temp.data +" ");
        temp = temp.next;
    }
    System.out.println();
}

public static void increment(Node<Integer> head){
    Node<Integer> temp = head;
    while(temp != null){
        temp.data++;
        temp = temp.next;
    }
}

public static void main(String args[]){

    Node<Integer> node1 = new Node<Integer>(10);
    Node<Integer> node2 = new Node<Integer>(20);
    node1.next = node2;
    increment(node1);
    print(node1);
   }
}
```

What will be the Output ?

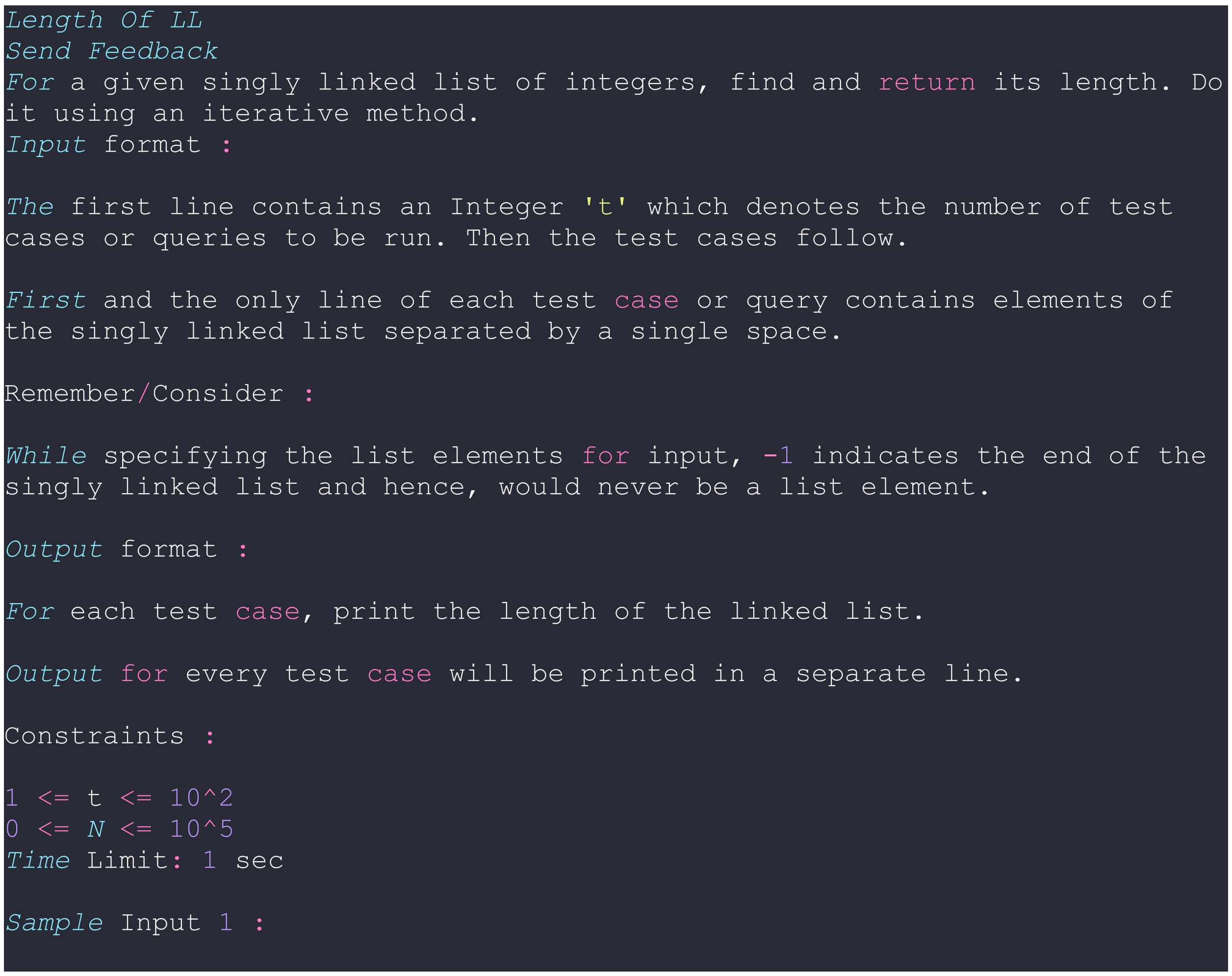


Options

This problem has only one correct answer

- 10 20
- [✓ 11 21]
- Error
- None of these

✓ Hurray! Correct Answer



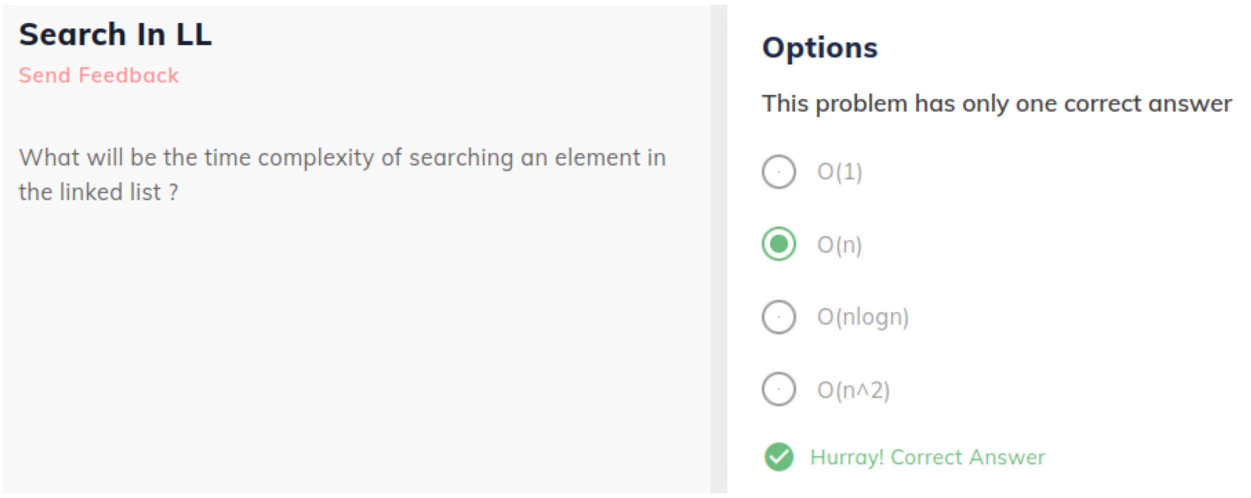
```
Length Of LL
Send Feedback
For a given singly linked list of integers, find and return its length. Do
it using an iterative method.
Input format :

The first line contains an Integer 't' which denotes the number of test
cases or queries to be run. Then the test cases follow.

First and the only line of each test case or query contains elements of
the singly linked list separated by a single space.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the
singly linked list and hence, would never be a list element.

Output format :

For each test case, print the length of the linked list.

Output for every test case will be printed in a separate line.

Constraints :

1 <= t <= 10^2
0 <= N <= 10^5
Time Limit: 1 sec

Sample Input 1 :
```

```
1
3 4 5 2 6 1 9 -1

Sample Output 1 :

7

Sample Input 2 :

2
10 76 39 -3 2 9 -23 9 -1
-1

Sample Output 2 :

8
0

public class Solution {

   public static int length(LinkedListNode<Integer> head){
       //Your code goes here
       LinkedListNode<Integer> temp = head;
       int count = 0;
       while(temp!=null){
           count++;
           temp = temp.next;
       }
       return count;
   }
}
```

```
Print ith Node
Send Feedback
For a given a singly linked list of integers and a position 'i', print the
node data at the 'i-th' position.
Note :

Assume that the Indexing for the singly linked list always starts from 0.

If the given position 'i',  is greater than the length of the given singly
linked list, then don't print anything.

Input format :

The first line contains an Integer 't' which denotes the number of test
cases or queries to be run. Then the test cases follow.

The first line of each test case or query contains the elements of the
singly linked list separated by a single space.

The second line contains the value of 'i'. It denotes the position in the
given singly linked list.
```

```
Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the
singly linked list and hence, would never be a list element.

Output format :

For each test case, print the node data at the 'i-th' position of the
linked list(if exists).

Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= N <= 10^5
i  >= 0
Time Limit: 1sec

Sample Input 1 :

1
3 4 5 2 6 1 9 -1
3

Sample Output 1 :

2

Sample Input 2 :

2
3 4 5 2 6 1 9 -1
0
9 8 4 0 7 8 -1
3

Sample Output 2 :

3
0

public class Solution {

   public static void printIthNode(LinkedListNode<Integer> head, int i){
       //Your code goes here
       if(i==0)
           return;
       int j = 0;
       LinkedListNode<Integer> temp = head;
       while(j<i){
           temp = temp.next;
           if(temp==null)
               return;
```

```
            j++;
        }
        System.out.println(temp.data);
    }
}
```



Search In LL

Send Feedback

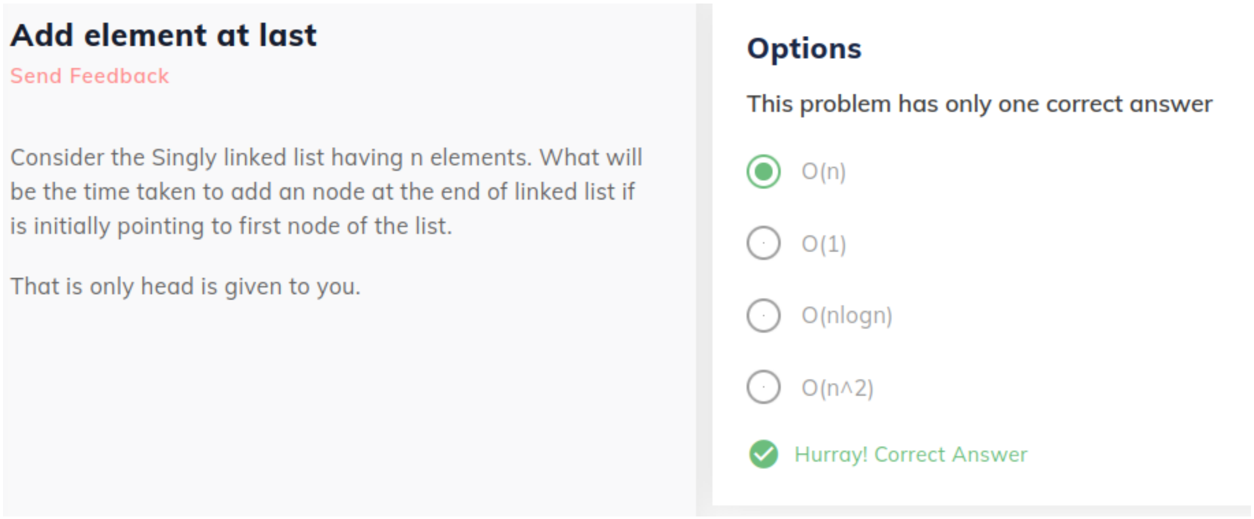
What will be the time complexity of searching an element in the linked list ?

Options

This problem has only one correct answer

- O(1)
- [✓ O(n)]
- O(nlogn)
- O(n^2)

Hurray! Correct Answer



Add element at last

Send Feedback

Consider the Singly linked list having n elements. What will be the time taken to add an node at the end of linked list if is initially pointing to first node of the list.

That is only head is given to you.

Options

This problem has only one correct answer

- [✓ O(n)]
- O(1)
- O(nlogn)
- O(n^2)

Hurray! Correct Answer

```
Insert Node In LL
Send Feedback
You have been given a singly linked list of integers, an integer value
called 'data' and a position with the name 'pos.'
Write a function to add a node to the list with the 'data' at the
specified position, 'pos.'
Note :

Assume that the Indexing for the singly linked list always starts from 0.

If the position is greater than the length of the singly linked list, you
should return the same linked list without any change.

Illustration :

The following images depict how the insertion has been taken place.
```

Image-I :

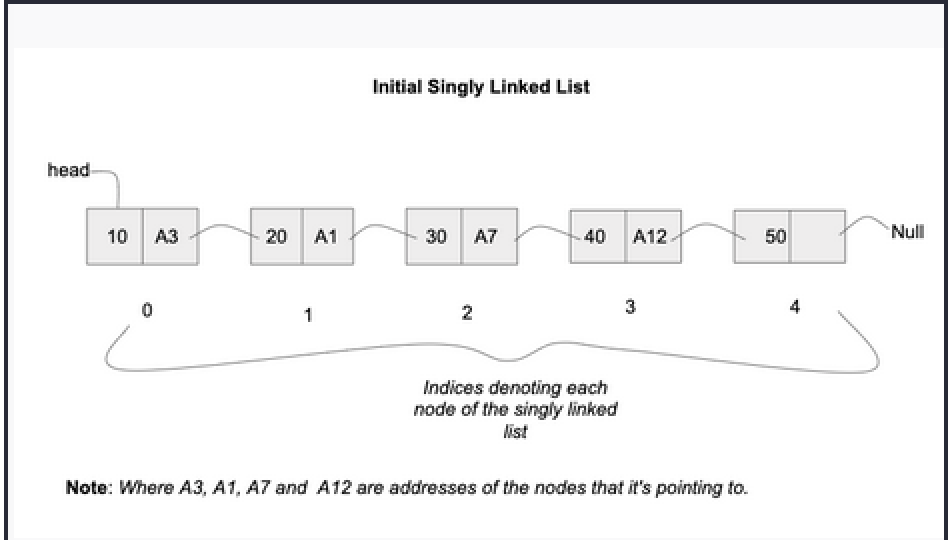

Initial Singly Linked List
head
10
A3
20
A1
30
A7
40
A12
50
Null
0
1
2
3
4
Indices denoting each node of the singly linked list
Note: Where A3, A1, A7 and A12 are addresses of the nodes that it's pointing to.

Image-II :

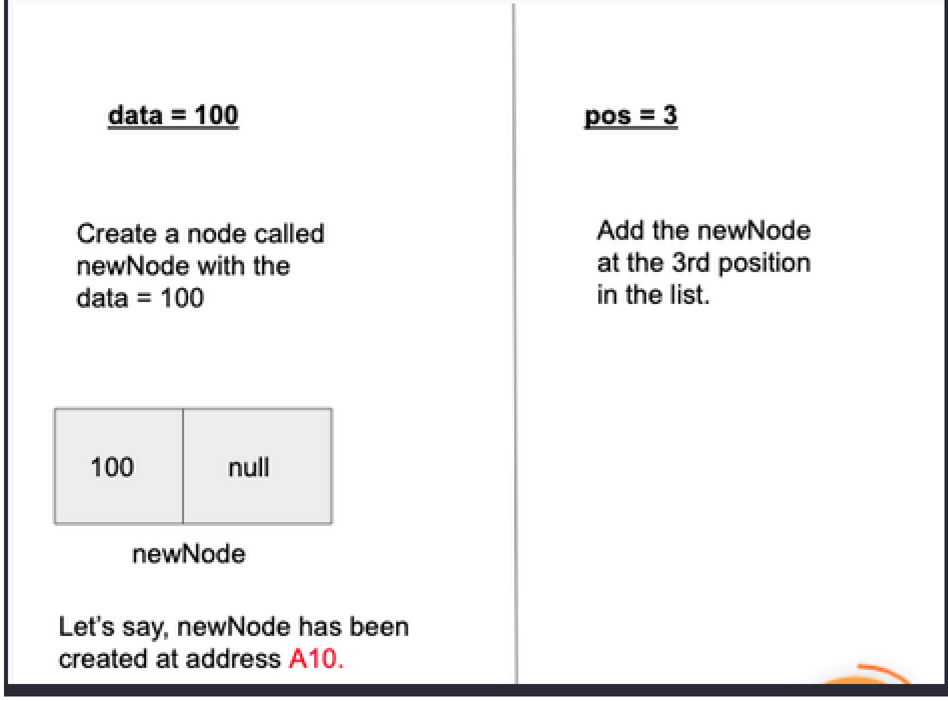

data = 100
pos = 3
Create a node called newNode with the data = 100
Add the newNode at the 3rd position in the list.
100
null
newNode
Let's say, newNode has been created at address A10.

Image-III :

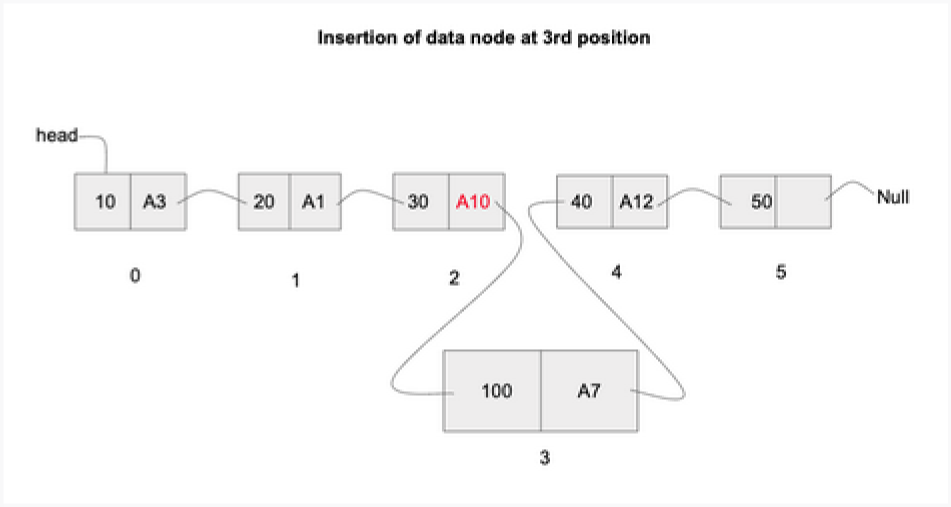
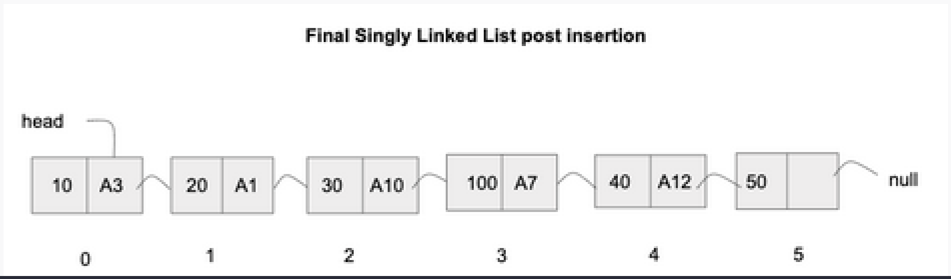


Image-IV :



The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

The first line of each test case or query contains the elements of the linked list separated by a single space.

The second line contains the two integer values of 'data' and 'pos' separated by a single space, respectively

Reminder/Consider :

While specifying the list elements for input, -1 indicates the end of the singly linked list and hence, would never be a list element.

Output format :

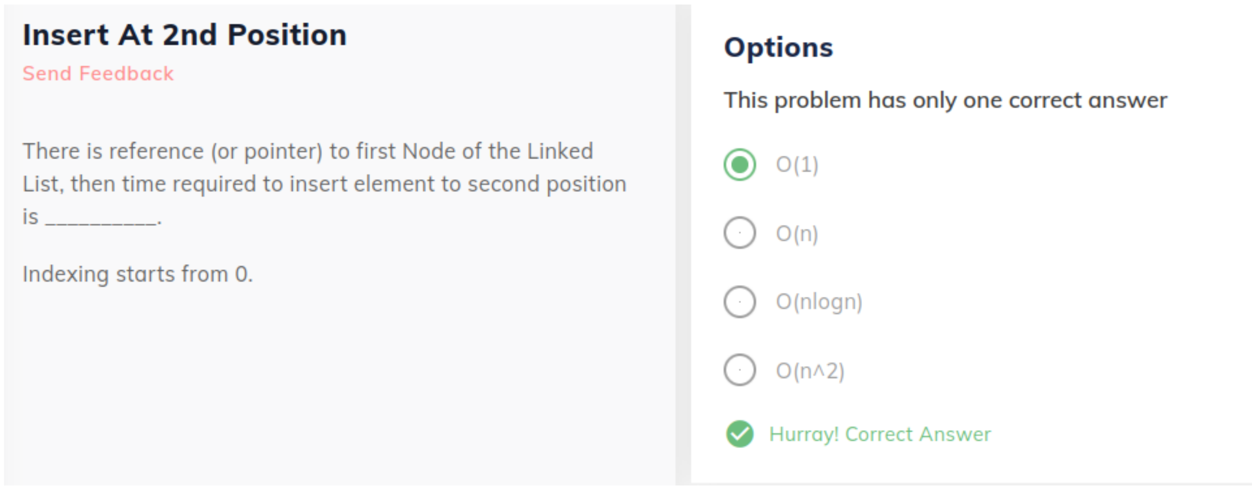
For each test case, print the resulting singly linked list of integers in a row, separated by a single space.

Output for every test case will be printed in a seperate line.

Constraints :

```
1 <= t <= 10^2
0 <= N <= 10^5
pos >= 0
Time Limit: 1sec

Sample Input 1 :

1
3 4 5 2 6 1 9 -1
3 100

Sample Output 1 :

3 4 5 100 2 6 1 9

Sample Input 2 :

2
3 4 5 2 6 1 9 -1
0 20
10 98 7 66 8 -1
5 99

Sample Output 2 :

20 3 4 5 2 6 1 9
10 98 7 66 8 99

public class Solution {

   public static LinkedListNode<Integer> insert(LinkedListNode<Integer>
head, int pos, int data){
       //Your code goes here
       LinkedListNode<Integer> newNode = new
LinkedListNode<Integer>(data);
       if(pos==0){
           newNode.next = head;
           head = newNode;
       }
       else{
           LinkedListNode<Integer> temp = head;
           int i = 0;
           while(i<pos-1 && temp!= null){

               temp=temp.next;
               i++;
           }
           if(temp !=null){

               newNode.next = temp.next;
               temp.next = newNode;
           }
       }
       return head;
```

```
    }
}
```



Insert At 2nd Position

Send Feedback

There is reference (or pointer) to first Node of the Linked List, then time required to insert element to second position is __________.

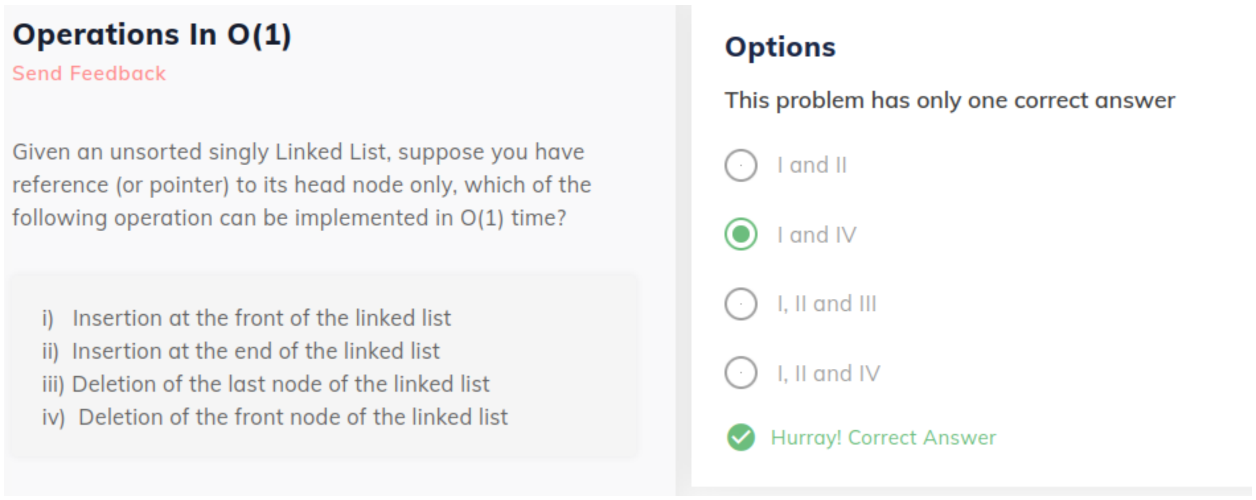
Indexing starts from 0.

Options

This problem has only one correct answer

- [✓ O(1)]
- O(n)
- O(nlogn)
- O(n^2)

✓ Hurray! Correct Answer



Operations In O(1)

Send Feedback

Given an unsorted singly Linked List, suppose you have reference (or pointer) to its head node only, which of the following operation can be implemented in O(1) time?

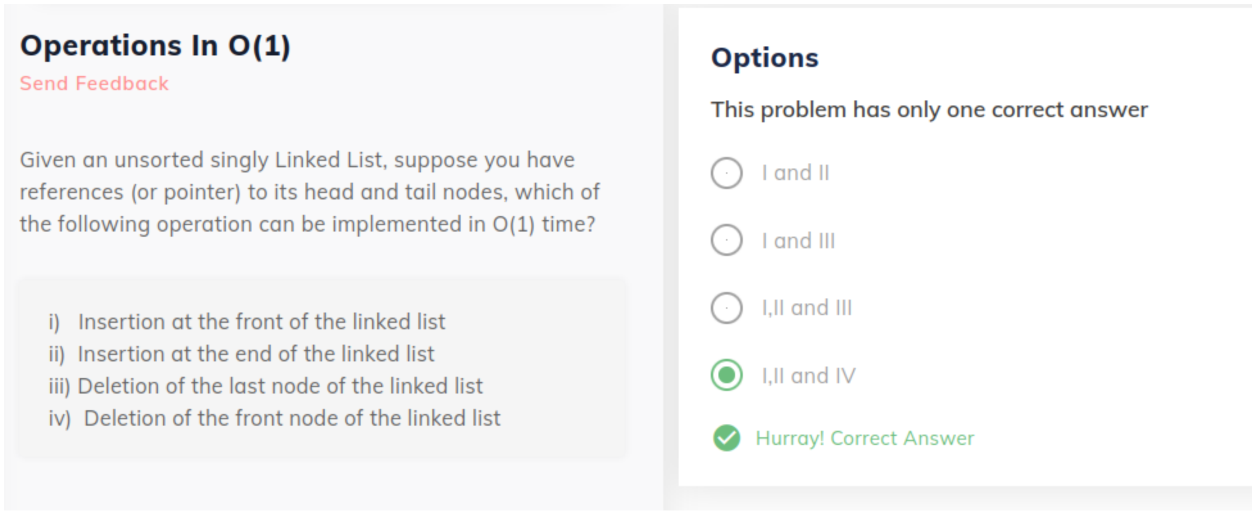
i) Insertion at the front of the linked list
ii) Insertion at the end of the linked list
iii) Deletion of the last node of the linked list
iv) Deletion of the front node of the linked list

Options

This problem has only one correct answer

- I and II
- [✓ I and IV]
- I, II and III
- I, II and IV

✓ Hurray! Correct Answer



Operations In O(1)

Send Feedback

Given an unsorted singly Linked List, suppose you have references (or pointer) to its head and tail nodes, which of the following operation can be implemented in O(1) time?

i) Insertion at the front of the linked list
ii) Insertion at the end of the linked list
iii) Deletion of the last node of the linked list
iv) Deletion of the front node of the linked list

Options

This problem has only one correct answer

- I and II
- I and III
- I,II and III
- [✓ I,II and IV]

✓ Hurray! Correct Answer

Delete Node In LL

```
Send Feedback
You have been given a linked list of integers. Your task is to write a
function that deletes a node from a given position, 'POS'.
Note :

Assume that the Indexing for the linked list always starts from 0.

If the position is greater than or equal to the length of the linked list,
you should return the same linked list without any change.

Illustration :

The following images depict how the deletion has been performed.

Image-I :
```

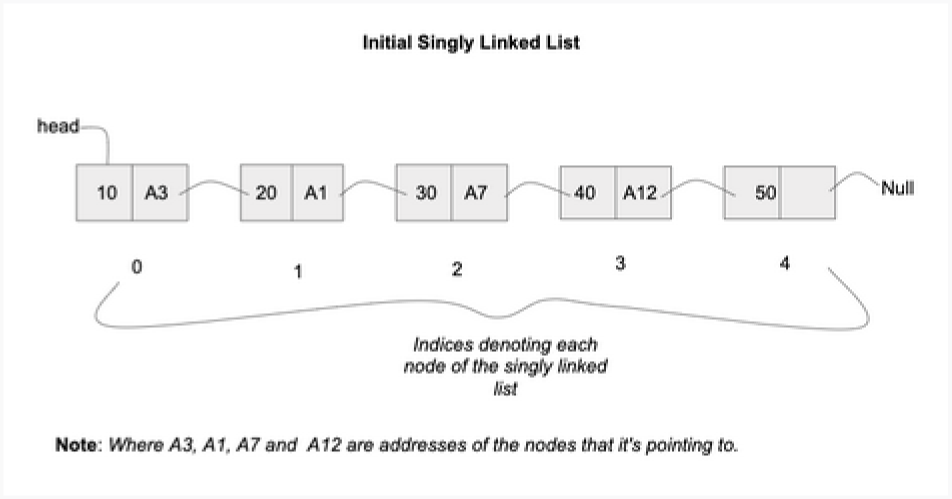
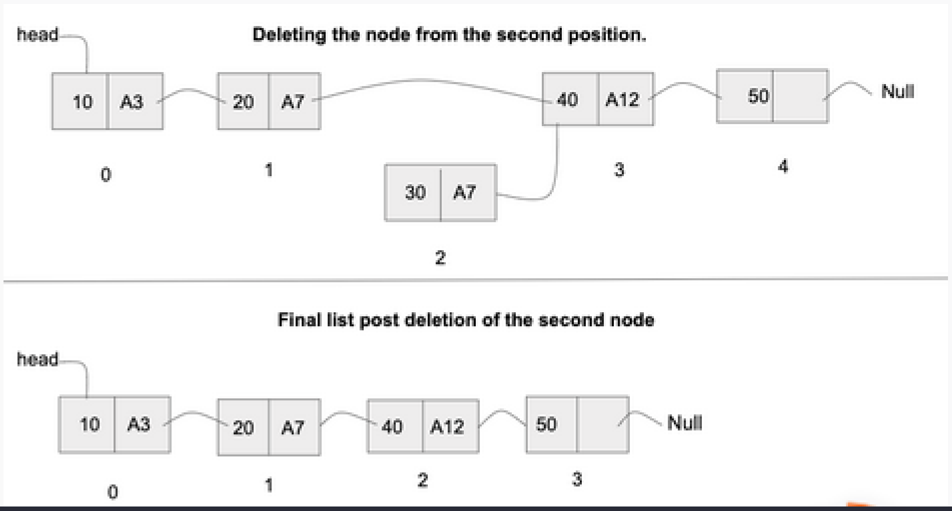

Initial Singly Linked List
head
10
A3
20
A1
30
A7
40
A12
50
Null
0
1
2
3
4
Indices denoting each node of the singly linked list
Note: Where A3, A1, A7 and A12 are addresses of the nodes that it's pointing to.

Image-II :


Deleting the node from the second position.
head
10
A3
20
A7
40
A12
50
Null
0
1
3
4
30
A7
2
Final list post deletion of the second node
head
10
A3
20
A7
40
A12
50
Null
0
1
2
3

```
The first line contains an Integer 'T' which denotes the number of test
cases or queries to be run. Then the test cases follow.

The first line of each test case or query contains the elements of the
linked list separated by a single space.

The second line of each test case contains the integer value of 'POS'. It
denotes the position in the linked list from where the node has to be
deleted.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the
singly linked list and hence, would never be a list element

Output format :

For each test case/query, print the resulting linked list of integers in a
row, separated by a single space.

Output for every test case will be printed in a separate line.

Note:

You are not required to print the output, it has already been taken care
of. Just implement the function.

Constraints :

1 <= T <= 10^2
0 <= N <= 10^5
POS >= 0

Time Limit: 1sec

Sample Input 1 :

1
3 4 5 2 6 1 9 -1
3

Sample Output 1 :

3 4 5 6 1 9

Sample Input 2 :

2
3 4 5 2 6 1 9 -1
0
10 20 30 40 50 60 -1
7

Sample Output 2 :
```

```
4 5 2 6 1 9
10 20 30 40 50 60

public class Solution {
   public static LinkedListNode<Integer>
deleteNode(LinkedListNode<Integer> head, int pos) {
       // Write your code here.

       if(pos==0){
           if(head!=null)
               head = head.next;
       }
       else {
           int count = 0;
           LinkedListNode<Integer> temp = head;
           while(count<pos-1 && temp.next!=null){
               temp = temp.next;
               count++;
           }
           if(temp.next==null){
               temp = null;
           }
           else{
               temp.next = temp.next.next;
           }
       }
       return head;
   }
}
```

```
Find a Node in Linked List
Send Feedback
You have been given a singly linked list of integers. Write a function
that returns the index/position of integer data denoted by 'N' (if it
exists). Return -1 otherwise.
Note :

Assume that the Indexing for the singly linked list always starts from 0.

Input format :

The first line contains an Integer 'T' which denotes the number of test
cases.

The first line of each test case or query contains the elements of the
singly linked list separated by a single space.
```

```
The second line contains the integer value 'N'. It denotes the data to be
searched in the given singly linked list.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the
singly linked list and hence -1 would never be a list element.

Output format :

For each test case, return the index/position of 'N' in the singly linked
list. Return -1, otherwise.

Output for every test case will be printed in a separate line.

Note:

You do not need to print anything; it has already been taken care of. Just
implement the given function.

Constraints :

1 <= T <= 10^2
0 <= M <= 10^5

Where 'M' is the size of the singly linked list.

Time Limit: 1 sec

Sample Input 1 :

2
3 4 5 2 6 1 9 -1
5
10 20 30 40 50 60 70 -1
6

Sample Output 1 :

2
```

```
-1

Explanation for Sample Output 1:

In test case 1, 'N' = 5 appears at position 2 (0-based indexing) in the
given linked list.

In test case 2, we can see that 'N' = 6 is not present in the given linked
list.

Sample Input 2 :

2
1 -1
2
3 4 5 2 6 1 9 -1
6

Sample Output 2 :

-1
4

Explanation for Sample Output 2:

In test case 1, we can see that 'N' = 2 is not present in the given linked
list.

In test case 2, 'N' = 6 appears at position 4 (0-based indexing) in the
given linked list.

public class Solution {
   public static int findNode(LinkedListNode<Integer> head, int n) {
        // Write your code here.
        if(head==null)
            return -1;
        LinkedListNode<Integer> temp = head;
        int count = 0;
        while(temp.data!=n){
            if(temp.next==null)
```

```
                return -1;
            temp = temp.next;
            count++;
        }
        return count;


    }
}
```

```
AppendLastNToFirst
Send Feedback
You have been given a singly linked list of integers along with an integer
'N'. Write a function to append the last 'N' nodes towards the front of
the singly linked list and returns the new head to the list.
Input format :

The first line contains an Integer 't' which denotes the number of test
cases or queries to be run. Then the test cases follow.

The first line of each test case or query contains the elements of the
singly linked list separated by a single space.

The second line contains the integer value 'N'. It denotes the number of
nodes to be moved from last to the front of the singly linked list.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the
singly linked list and hence, would never be a list element.

Output format :

For each test case/query, print the resulting singly linked list of
integers in a row, separated by a single space.

Output for every test case will be printed in a seperate line.

Constraints :

```

```
1 <= t <= 10^2
0 <= M <= 10^5
0 <= N < M
Time Limit: 1sec

Where 'M' is the size of the singly linked list.

Sample Input 1 :

2
1 2 3 4 5 -1
3
10 20 30 40 50 60 -1
5

Sample Output 1 :

3 4 5 1 2
20 30 40 50 60 10

Sample Input 2 :

1
10 6 77 90 61 67 100 -1
4

Sample Output 2 :

90 61 67 100 10 6 77

Explanation to Sample Input 2 :

We have been required to move the last 4 nodes to the front of the list.
Here, "90->61->67->100" is the list which represents the last 4 nodes.
When we move this list to the front then the remaining part of the initial
list which is, "10->6->77" is attached after 100. Hence, the new list
formed with an updated head pointing to 90.

public class Solution {
   public static LinkedListNode<Integer>
appendLastNToFirst(LinkedListNode<Integer> head, int n) {
       //Your code goes here
       if(head==null || n==0)
           return head;

       LinkedListNode<Integer> temp = head;
       int count = 0;

       while(temp.next!=null){
           temp = temp.next;
           count++;
       }
       temp.next=head;
       count = count-n;
       int i = 0;
       temp = head;
       while(i<count){
           temp = temp.next;
           i++;
       }
       head = temp.next;
       temp.next = null;
       return head;
   }
}
```

```
Eliminate duplicates from LL
Send Feedback
You have been given a singly linked list of integers where the elements
are sorted in ascending order. Write a function that removes the
consecutive duplicate values such that the given list only contains unique
elements and returns the head to the updated list.
Input format :

The first line contains an Integer 't' which denotes the number of test
cases or queries to be run. Then the test cases follow.
```

```
The first and the only line of each test case or query contains the
elements(in ascending order) of the singly linked list separated by a
single space.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the
singly linked list and hence, would never be a list element.

Output format :

For each test case/query, print the resulting singly linked list of
integers in a row, separated by a single space.

Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= M <= 10^5
Time Limit: 1sec

Where 'M' is the size of the singly linked list.

Sample Input 1 :

1
1 2 3 3 3 3 4 4 4 5 5 7 -1

Sample Output 1 :

1 2 3 4 5 7

Sample Input 2 :

2
10 20 30 40 50 -1
10 10 10 10 -1

Sample Output 2 :
```

```
10 20 30 40 50
10

public class Solution {

   public static LinkedListNode<Integer>
removeDuplicates(LinkedListNode<Integer> head) {
        //Your code goes here

        if(head!=null)
        {
             LinkedListNode<Integer> temp = head;
             LinkedListNode<Integer> temp1 = head.next;
             while(temp1!=null){
                  if(!temp1.data.equals(temp.data) ){

                      //
-------------------------------------------------------
                      // .equals cleared all test cases while "=" cleared
only 3 test cases
                      //
-------------------------------------------------------
                      temp.next = temp1;
                      temp = temp.next;
                  }
                  temp1 = temp1.next;
             }
             temp.next = null;

        }
        return head;
   }
}
```

Print Reverse LinkedList

Send Feedback

You have been given a singly linked list of integers. Write a function to print the list in a reverse order.

```
To explain it further, you need to start printing the data from the tail
and move towards the head of the list, printing the head data at the end.
Note :

You can’t change any of the pointers in the linked list, just print it in
the reverse order.

Input format :

The first line contains an Integer 't' which denotes the number of test
cases or queries to be run. Then the test cases follow.

The first and the only line of each test case or query contains the
elements of the singly linked list separated by a single space.

Remember/Constraints :

While specifying the list elements for input, -1 indicates the end of the
singly linked list and hence, would never be a list element.

Output format :

For each test case, print the singly linked list of integers in a reverse
fashion, in a row, separated by a single space.

Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= M <= 10^3
Time Limit: 1sec

Where 'M' is the size of the singly linked list.

Sample Input 1 :

1
1 2 3 4 5 -1
```

```
Sample Output 1 :

5 4 3 2 1

Sample Input 2 :

2
1 2 3 -1
10 20 30 40 50 -1

Sample Output 2 :

3 2 1
50 40 30 20 10

public class Solution {

   public static void printReverse(LinkedListNode<Integer> root) {
       //Your code goes here

       if(root==null)
           return;
       else
           printReverse(root.next);
       System.out.print(root.data+" ");



   }
}
```

```
Palindrome LinkedList
Send Feedback
You have been given a head to a singly linked list of integers. Write a
function check to whether the list given is a 'Palindrome' or not.
Input format :

The first line contains an Integer 't' which denotes the number of test
cases or queries to be run. Then the test cases follow.
```

```
First and the only line of each test case or query contains the the
elements of the singly linked list separated by a single space.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the
singly linked list and hence, would never be a list element.

Output format :

For each test case, the only line of output that print 'true' if the list
is Palindrome or 'false' otherwise.

Constraints :

1 <= t <= 10^2
0 <= M <= 10^5
Time Limit: 1sec

Where 'M' is the size of the singly linked list.

Sample Input 1 :

1
9 2 3 3 2 9 -1

Sample Output 1 :

true

Sample Input 2 :

2
0 2 3 2 5 -1
-1

Sample Output 2 :

false
true
```

```
Explanation for the Sample Input 2 :

For the first query, it is pretty intuitive that the the given list is not
a palindrome, hence the output is 'false'.

For the second query, the list is empty. An empty list is always a
palindrome , hence the output is 'true'.

public class Solution {

    static LinkedListNode<Integer> reverse(LinkedListNode<Integer> node)
    {
        LinkedListNode<Integer> prev = null;
        LinkedListNode<Integer> current = node;
        LinkedListNode<Integer> next = null;
        while (current != null) {
            next = current.next;
            current.next = prev;
            prev = current;
            current = next;
        }
        node = prev;
        return node;
    }

    public static boolean isPalindrome(LinkedListNode<Integer> head) {
        //Your code goes here
        if(head==null)
            return true;
        LinkedListNode<Integer> temp = head;
        int length = 0;
        while(temp!=null){
            length++;
            temp = temp.next;
        }
        if(length>1){
            temp = head;
            int count = 0;
            while(count<length/2){
                temp = temp.next;
                count++;
            }
            LinkedListNode<Integer> secondList;
            if(length%2==0)
                secondList = temp.next;
            else
                secondList = temp.next.next;
            secondList = reverse(temp);
            count = 0;
            temp = head;
            while(count<length && temp!=null && secondList!=null){
                // System.out.println(temp.data+" "+secondList.data);
                if(temp.data!=secondList.data)
                {
                    // System.out.println(temp.data+" "+secondList.data);
                    return false;
                }
                temp = temp.next;
                secondList = secondList.next;
                count++;
            }
        }
        return true;

    }
}
```