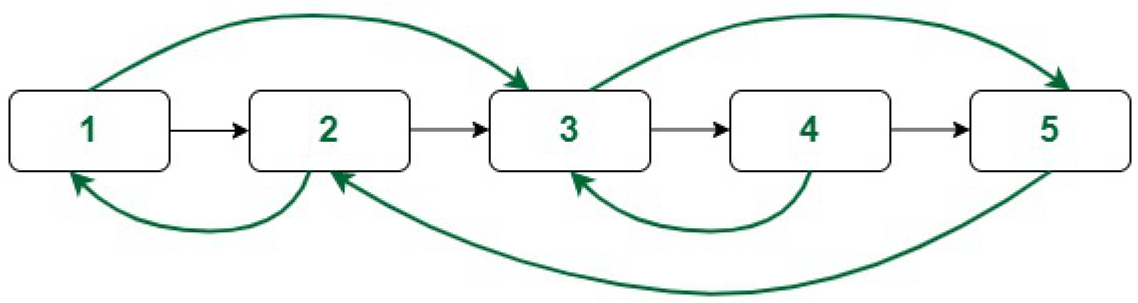
Clone a linked list with Random Pointer

An example of linked list with a random pointerGiven a **linked list** of size **N** where each node has two links: one pointer points to the next node and the second pointer points to any node in the list. The task is to create a clone of this linked list in **O(N) time**.

Note: The pointer pointing to the next node is '**next**' pointer and the one pointing to an arbitrary node is called **'arbit' pointer** as it can point to any arbitrary node in the linked list.

An example of the linked list is shown in the below image:



An example of linked list with a random pointerAn example of linked list with a random pointer

Approach 1 (Using Extra Space): The idea to solve this problem is:

> *First create a single linked list with only the 'next' pointer and use a mapping to map the new nodes to their corresponding nodes in the given linked list. Now use this mapping to point the arbitrary node from any node in the newly created list.*

Follow the steps mentioned beloved to implement the above idea:

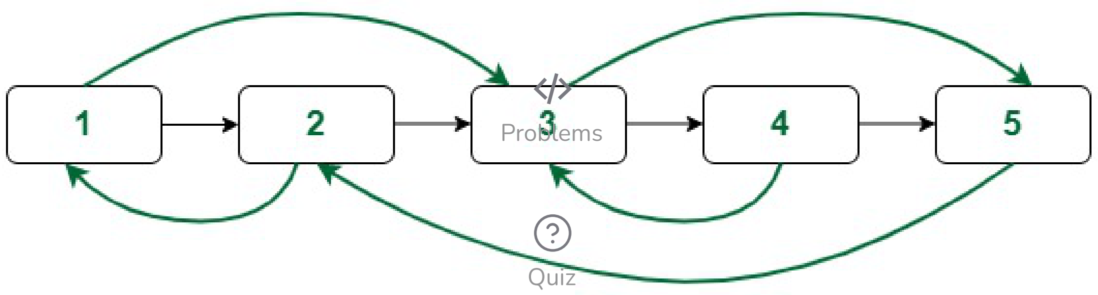
- Create a duplicate (say **Y**) for each node (say **X**) and map them with corresponding old nodes (say **mp**, So **mp[X] = Y**).
- Create the single linked list of the duplicate nodes where each node only has the **'next'** pointer.
- Now iterate over the old linked list and do the following:

- Make the **arbit** pointer of the duplicate node pointing to the duplicate of

Follow the illustration below for a better understanding:

Illustration:

Consider the linked list shown below:

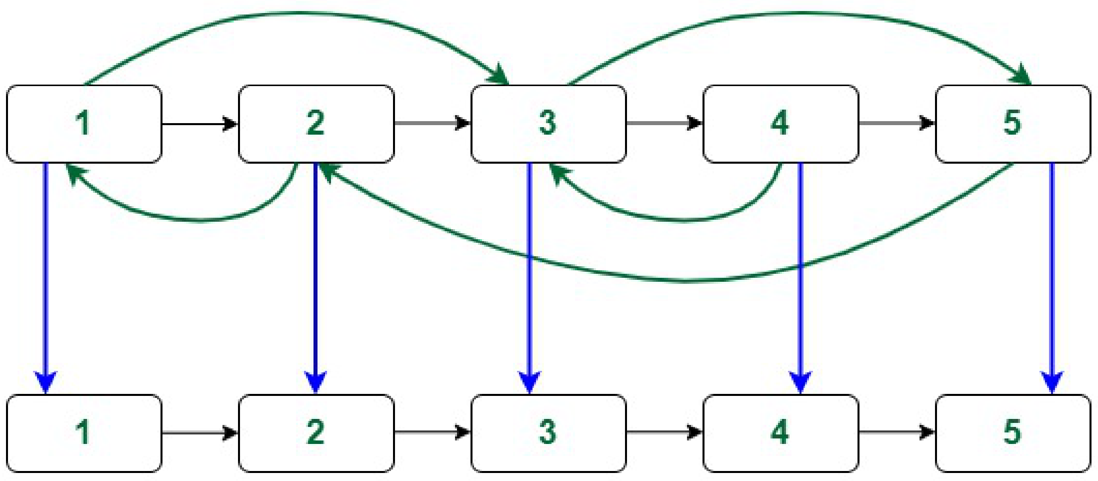


Original linked list

The green links are the arbit pointers

Creating copy of Nodes and next pointer:

Initially create single linked list of duplicate nodes with only the next pointers and map them with the old ones.
Here the blue coloured links are used to show the mapping.



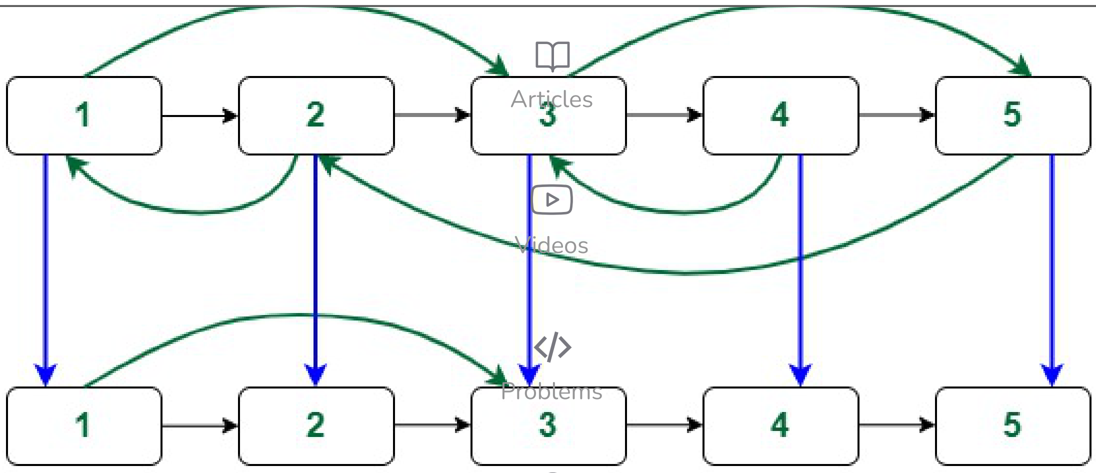
New linked list mapped with old nodes

Menu

Linking the arbit pointers:

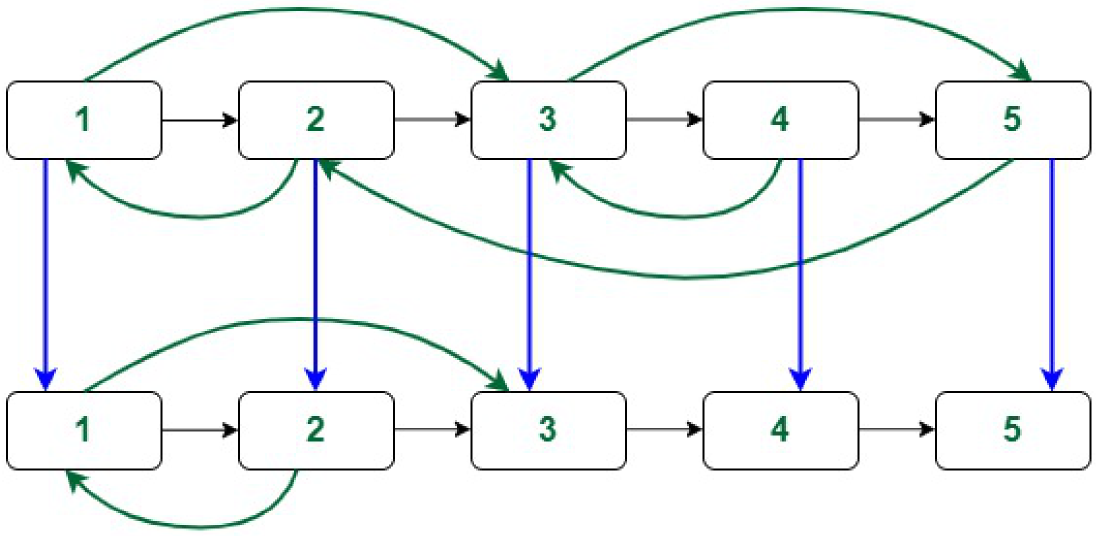
Track Progress
99 of **132** Complete. (75%)

pointers.



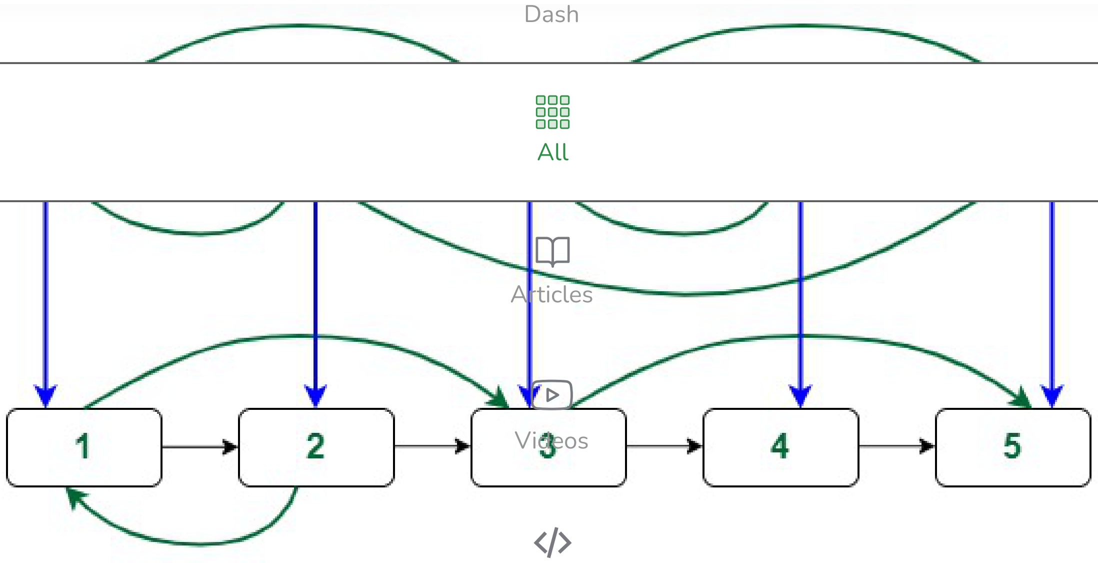
Linking arbit pointer of duplicate of 1st node

At second node:



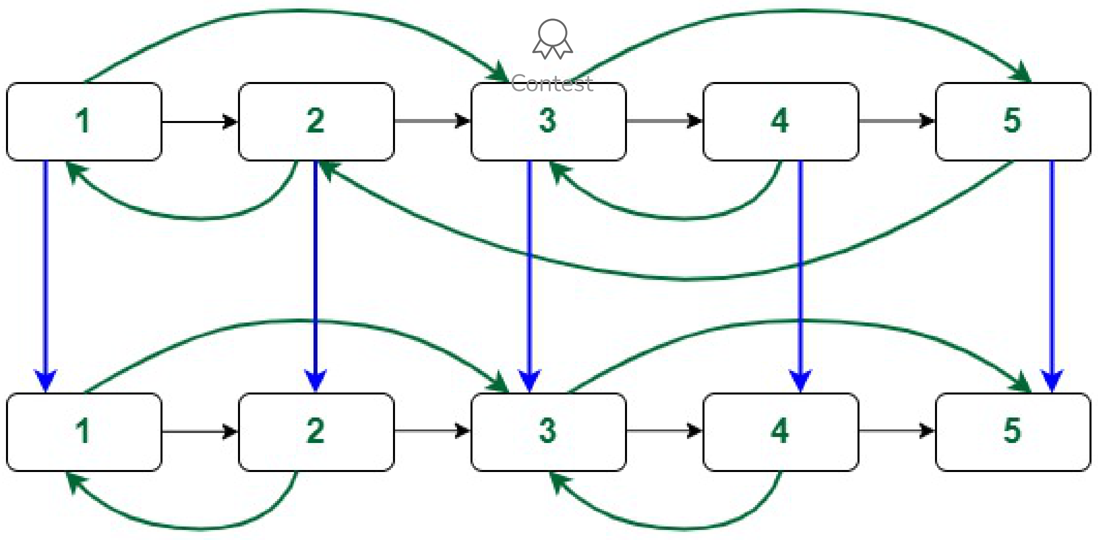
Linking arbit pointer of duplicate of 2nd node

At third node:



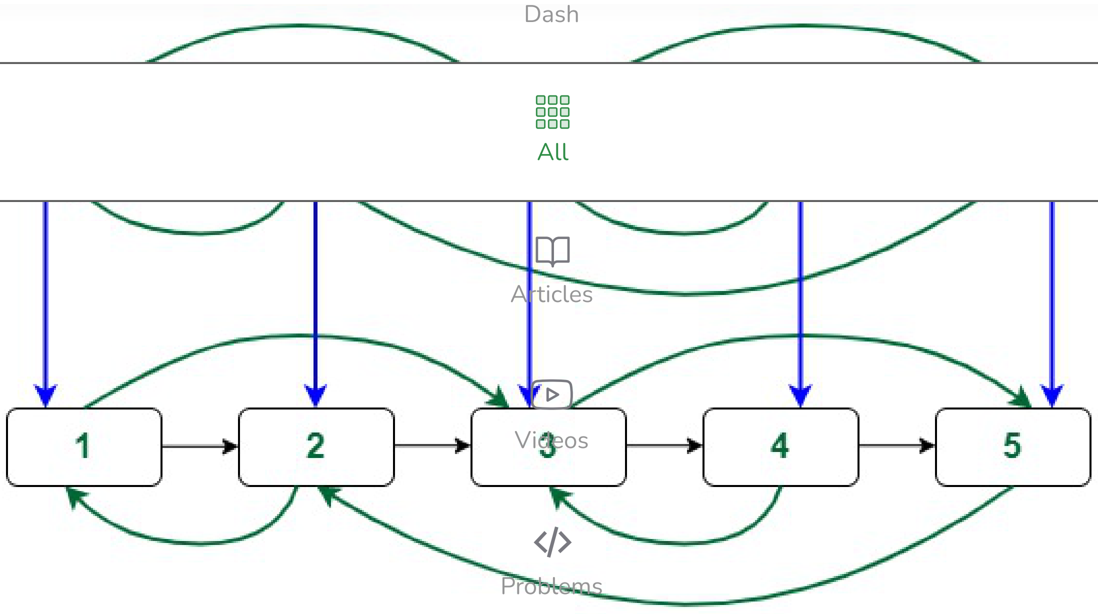
Linking arbit pointer of duplicate of 3rd node

At fourth node:



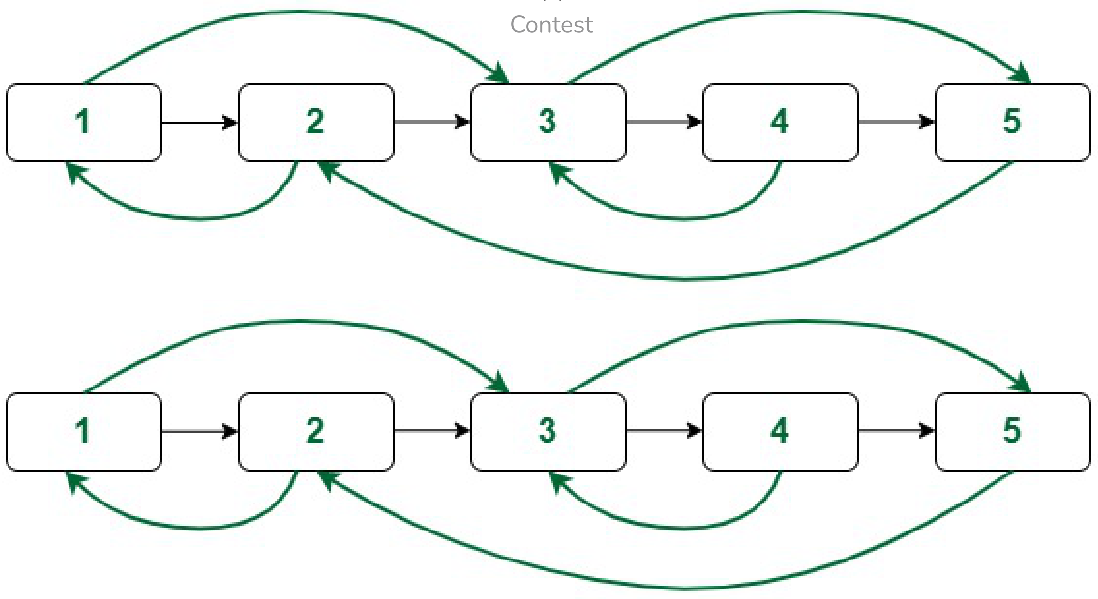
Linking arbit pointer of duplicate of 4th node

At fifth node:



Linking arbit pointer of duplicate of 5th node

The final linked list is as shown below:



The original and the clone

Below is the implementation of the above approach:

C++

```
// C++ code to implement the approach

#include <bits/stdc++.h>
```

```
// Structure of a node of linked list
```

```
    Node* arbit;

    // Constructor
    Node(int x)
    {
        this->val = x;
        this->next = NULL;
        this->arbit = NULL;
    }
};

// Function to clone the linked list
Node* cloneLinkedList(Node* head)
{
    // Map to store the mapping of
    // old nodes with new ones
    unordered_map<Node*, Node*> mp;
    Node *temp, *nhead;

    // Duplicate of the first node
    temp = head;
    nhead = new Node(temp->val);
    mp[temp] = nhead;

    // Loop to create duplicates of nodes
    // with only next pointer
    while (temp->next != NULL) {
        nhead->next
            = new Node(temp->next->val);
        temp = temp->next;
        nhead = nhead->next;
        mp[temp] = nhead;
    }
    temp = head;

    // Loop to clone the arbit pointers
    while (temp != NULL) {
```

Dash

All

Articles

Videos

Problems

Quiz

Contest

```
        }
    }

    // Function to print the linked list
    void printList(Node* head)
    {
        cout << head->val << "("
            << head->arbit->val << ")";
        head = head->next;
        while (head != NULL) {
            cout << " -> " << head->val << "("
                << head->arbit->val << ")";
            head = head->next;
        }
        cout << endl;
    }

    // Driver code
    int main()
    {
        // Creating a linked list with random pointer
        Node* head = new Node(1);
        head->next = new Node(2);
        head->next->next = new Node(3);
        head->next->next->next = new Node(4);
        head->next->next->next->next
            = new Node(5);
        head->arbit = head->next->next;
        head->next->arbit = head;
        head->next->next->arbit
            = head->next->next->next->next;
        head->next->next->next->arbit
            = head->next->next;
        head->next->next->next->next->arbit
            = head->next;

        // Print the original list
        cout << "The original linked list:\n";
```

Menu

Track Progress

99 of **132** Complete. (75%)

```
    // Function call
```

Dash

All

```
    printList(sol);

    return 0;
}
```

Articles

Videos

Problems

Quiz

Contest

Time Complexity: O(N)
Auxiliary Space: O(N)

Approach 2 (Without Using Extra Space):

- Create the copy of node 1 and insert it between node 1 & node 2 in the original Linked List, create a copy of 2 and insert it between 2 & 3. Continue in this fashion, add the copy of N after the Nth node
- Now copy the random link in this fashion

```
 original->next->random= original->random->next;  /*TRAVERSE
TWO NODES*/
```

- This works because original->next is nothing but a copy of the original and Original->random->next is nothing but a copy of the random.

- Now restore the original and copy linked lists in this fashion in a single loop.

```
original->next = original->next->next;
     copy->next = copy->next->next;
```

- Ensure that original->next is NULL and return the cloned list

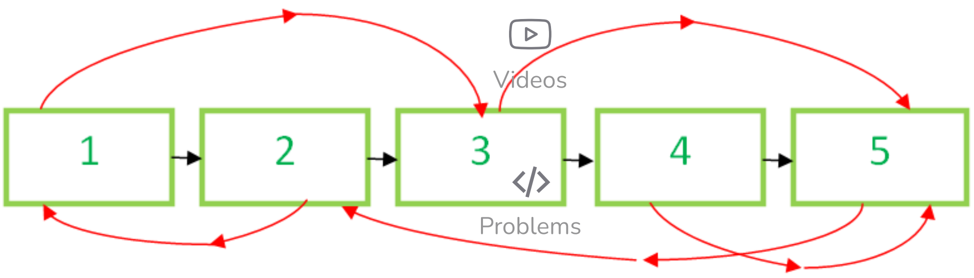
Menu

Track Progress
99 of **132** Complete. (75%)

Dash

All

Articles



Videos

Problems

Quiz

Below is the implementation.

C++ Java

Contest

```
// Java program to clone a linked list with next
// and arbit pointers in O(n) time
class GfG {

    // Structure of linked list Node
    static class Node {
        int data;
        Node next, random;
        Node(int x)
        {
            data = x;
            next = random = null;
        }
    }

    // Utility function to print the list.
    static void print(Node start)
    {
        Node ptr = start;
        while (ptr != null) {
            System.out.println("Data = " + ptr.data
                               + ", Random = "
```

Menu

Track Progress

99 of **132** Complete. (75%)

Dash

All

Articles

Videos

Problems

Quiz

Contest

```
        }

    // linked list in O(1) space
    static Node clone(Node start)
    {
        Node curr = start, temp = null;

        // insert additional node after
        // every node of original list
        while (curr != null) {
            temp = curr.next;

            // Inserting node
            curr.next = new Node(curr.data);
            curr.next.next = temp;
            curr = temp;
        }
        curr = start;

        // adjust the random pointers of the
        // newly added nodes
        while (curr != null) {
            if (curr.next != null)
                curr.next.random = (curr.random != null)
                                   ? curr.random.next
                                   : curr.random;

            // move to the next newly added node by
            // skipping an original node
            curr = curr.next.next;
        }

        Node original = start, copy = start.next;

        // save the start of copied linked list
        temp = copy;

        // now separate the original list and copied list
        while (original != null) {
```

Menu

Get 90% Refund!

Courses

Tutorials

```
        copy.next = (copy.next != null) ? copy.next.next
```

```
        }
        return temp;
    }

    // Driver code
    public static void main(String[] args)
    {
        Node start = new Node(1);
        start.next = new Node(2);
        start.next.next = new Node(3);
        start.next.next.next = new Node(4);
        start.next.next.next.next = new Node(5);

        // 1's random points to 3
        start.random = start.next.next;

        // 2's random points to 1
        start.next.random = start;

        // 3's and 4's random points to 5
        start.next.next.random = start.next.next.next.next;
        start.next.next.next.random
            = start.next.next.next.next;

        // 5's random points to 2
        start.next.next.next.next.random = start.next;

        System.out.println("Original list : ");
        print(start);

        System.out.println("Cloned list : ");
        Node cloned_list = clone(start);
        print(cloned_list);
    }
}
```

Dash

```
Original list :
```

All

```
Data = 4, Random  = 5
Data = 5, Random  = 2

Cloned list :
Data = 1, Random  = 3
Data = 2, Random  = 1
Data = 3, Random  = 5
Data = 4, Random  = 5
Data = 5, Random  = 2
```

Articles

Videos

Problems

Time Complexity: O(n) As we are moving through the list thrice, i.e. 3n , but in asymptotic notations we drop the constant terms
Auxiliary Space: O(1) As no extra space is used. The n nodes which are inserted in between the nodes was already required to clone the list, so we can say that we did not use any extra space.

Quiz

Contest

Mark as Read

Report An Issue

If you are facing any issue on this page. Please let us know.

Menu

Track Progress

99 of **132** Complete. (75%)