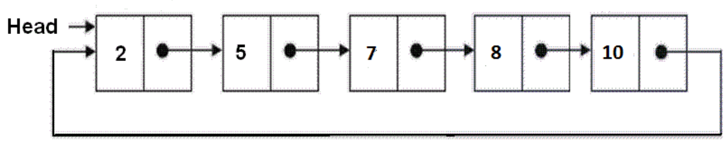
Check if a linked list is Circular Linked List

Given a singly linked list, find if the linked list is circular or not. A linked list is called circular if it not NULL terminated and all nodes are connected in the form of a cycle. Below is an example of circular linked list.



An empty linked list is considered as circular.

Note that this problem is different from cycle detection problem, here all nodes have to be part of cycle.

The idea is to store head of the linked list and traverse it. If we reach NULL, linked list is not circular. If reach head again, linked list is circular.

```
// C++ program to check if linked list is circular
#include<bits/stdc++.h>
using namespace std;

/* Link list Node */
struct Node
{
   int data;
   struct Node* next;
};

/* This function returns true if given linked
  list is circular, else false. */
bool isCircular(struct Node *head)
{
   // An empty linked list is circular
   if (head == NULL)
      return true;

   // Next of head
   struct Node *node = head->next;

   // This loop would stope in both cases (1) If
   // Circular (2) Not circular
   while (node != NULL && node != head)
      node = node->next;

   // If loop stopped because of circular
   // condition
   return (node == head);
}

// Utility function to create a new node.
Node *newNode(int data)
{
   struct Node *temp = new Node;
   temp->data = data;
   temp->next = NULL;
   return temp;
}

/* Drier program to test above function*/
int main()
{
   /* Start with the empty list */
   struct Node* head = newNode(1);
   head->next = newNode(2);
   head->next->next = newNode(3);
   head->next->next->next = newNode(4);

   isCircular(head)? cout << "Yes\n" :
                cout << "No\n" ;

   // Making linked list circular
   head->next->next->next->next = head;

   isCircular(head)? cout << "Yes\n" :
                cout << "No\n" ;

   return 0;
}
```

Output :

```
No
Yes
```