



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment- 1.3

**Student Name:** UTKARSH JOSHI

**UID:** 21BCS9158

**Branch:** CSE-Gen

**Section/Group:** ST 802-A

**Semester:** 5<sup>th</sup>

**Date of Performance:** 17/08/23

**Subject Name:** Advanced Programming

**Subject Code:** 21CSP-314

### **1. Aim:**

Solve the following problems on hackerrank:

- 1- Cycle Detection
- 2- Compare two linked lists

**2. Objective:** To perform different operation in LinkedList.

### **3. Code :**

#### **Program -1**

```
#!/bin/python3
```

```
import math  
import os  
import random  
import re  
import sys
```

```
class SinglyLinkedListNode:
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
def __init__(self, node_data):  
    self.data = node_data  
    self.next = None
```

```
class SinglyLinkedList:
```

```
    def __init__(self):  
        self.head = None  
        self.tail = None
```

```
    def insert_node(self, node_data):  
        node = SinglyLinkedListNode(node_data)
```

```
        if not self.head:  
            self.head = node  
        else:  
            self.tail.next = node
```

```
        self.tail = node
```

```
def print_singly_linked_list(node, sep, fptr):
```

```
    while node:  
        fptr.write(str(node.data))
```

```
        node = node.next
```

```
    if node:  
        fptr.write(sep)
```

# Complete the has\_cycle function below.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
#
# For your reference:
#
# SinglyLinkedListNode:
#     int data
#     SinglyLinkedListNode next
#
#
def has_cycle(head):
    #intialize two pointers
    slow = fast = head

    #main logic
    while fast != None and fast.next != None:
        slow = slow.next
        fast = fast.next.next

    #check if both pointers are some
    if slow == fast:
        return True

    return False

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    tests = int(input())

    for tests_itr in range(tests):
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
index = int(input())

llist_count = int(input())

llist = SinglyLinkedList()

for _ in range(llist_count):
    llist_item = int(input())
    llist.insert_node(llist_item)

extra = SinglyLinkedListNode(-1);
temp = llist.head;

for i in range(llist_count):
    if i == index:
        extra = temp

    if i != llist_count-1:
        temp = temp.next

temp.next = extra

result = has_cycle(llist.head)

fptr.write(str(int(result)) + '\n')

fptr.close()
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Program -2

```
#!/bin/python3
```

```
import os
```

```
import sys
```

```
class SinglyLinkedListNode:
```

```
    def __init__(self, node_data):
```

```
        self.data = node_data
```

```
        self.next = None
```

```
class SinglyLinkedList:
```

```
    def __init__(self):
```

```
        self.head = None
```

```
        self.tail = None
```

```
    def insert_node(self, node_data):
```

```
        node = SinglyLinkedListNode(node_data)
```

```
        if not self.head:
```

```
            self.head = node
```

```
        else:
```

```
            self.tail.next = node
```

```
        self.tail = node
```

```
def print_singly_linked_list(node, sep, fptr):
```

```
    while node:
```

```
        fptr.write(str(node.data))
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
node = node.next
```

```
if node:
```

```
    fptr.write(sep)
```

```
# Complete the compare_lists function below.
```

```
#
```

```
# For your reference:
```

```
#
```

```
# SinglyLinkedListNode:
```

```
#     int data
```

```
#     SinglyLinkedListNode next
```

```
#
```

```
#
```

```
def compare_lists(llist1, llist2):
```

```
    head1 = llist1
```

```
    head2 = llist2
```

```
    #while head != None and head2 != None:
```

```
    while head1 and head2:
```

```
        #check data are same
```

```
        if head1.data == head2.data:
```

```
            head1 = head1.next
```

```
            head2 = head2.next
```

```
        else:
```

```
            return 0
```

```
    if head1 == None and head2 == None:
```

```
        return 1
```

```
    else:
```

```
        return 0
```

```
if __name__ == '__main__':
```

```
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
tests = int(input())

for tests_itr in range(tests):
    llist1_count = int(input())

    llist1 = SinglyLinkedList()

    for _ in range(llist1_count):
        llist1_item = int(input())
        llist1.insert_node(llist1_item)

    llist2_count = int(input())

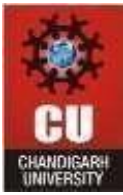
    llist2 = SinglyLinkedList()

    for _ in range(llist2_count):
        llist2_item = int(input())
        llist2.insert_node(llist2_item)

    result = compare_lists(llist1.head, llist2.head)

    fptr.write(str(int(result)) + '\n')

fptr.close()
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 5. Output:

### Program 1:

The screenshot shows a web browser displaying a Hackerrank challenge page. The URL is <https://www.hackerrank.com/challenges/three-month-preparation-kit-detect-whether-a-linked-list-contains-a-cycle/problem...>. The code editor shows the following Python code:

```
80 extra = SinglyLinkedListNode(-1);
81 temp = llist.head;
82
83 for i in range(llist_count):
84     # Add node to the end of the list
```

Line: 62 Col: 17

Buttons: Upload Code as File, Test against custom input, Run Code, Submit Code

Test cases (all passed):

- Test case 0
- Test case 1
- Test case 2
- Test case 3
- Test case 4
- Test case 5
- Test case 6

Compiler Message: Success

Input (stdin):

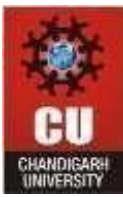
```
1 1
2 -1
3 1
4 1
```

Expected Output:

```
1 0
```

Footer: Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy





# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Program 2:

The screenshot shows the HackerRank interface for the 'Compare two linked lists' challenge. The problem description states: 'You're given the pointer to the head nodes of two linked lists. Compare the data in the nodes of the linked lists to check if they are equal. If all data attributes are equal and the lists are the same length, return 1. Otherwise, return 0.' Example lists are provided: `l1 = 1 → 2 → 3 → NULL` and `l2 = 1 → 2 → 3 → 4 → NULL`. The function signature is `compare_lists(singlyLinkedListNode l1, singlyLinkedListNode l2)`. The test cases table shows 6 cases, all passed.

Test Case	Input	Output
Test case 0	1 2	1
Test case 1	2 2	1
Test case 2	3 1	0
Test case 3	4 2	0
Test case 4	5 1	0
Test case 5	6 1	0
Test case 6	7 2	0

## Learning outcomes:

1. Understand what a linked list is and its structure.
2. Differentiate between arrays and linked lists as data structures.
3. Recognize the applications of linked lists in computer science and the real world.
4. Understand the cost/benefit trade-offs of using arrays vs linked lists.

