Experiment- 1.3

Student Name: UTKARSH JOSHI
UID: 21BCS9158
Branch: CSE-Gen
Section/Group: ST 802-A
Semester: 5th
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:
```

```
    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.
```

```
#
# 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):
```

```
    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()
```

Program -2

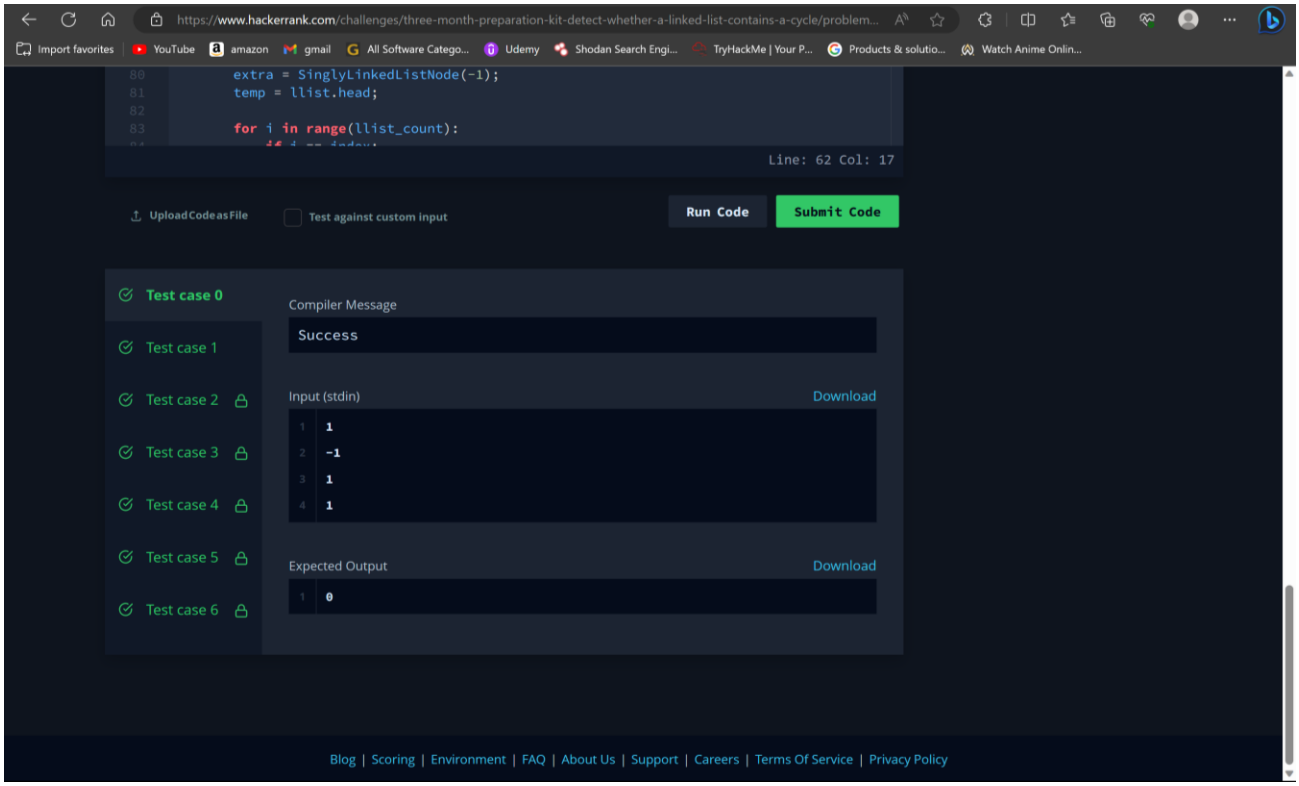
```
#!/bin/python3

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))
```

```
        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')
```

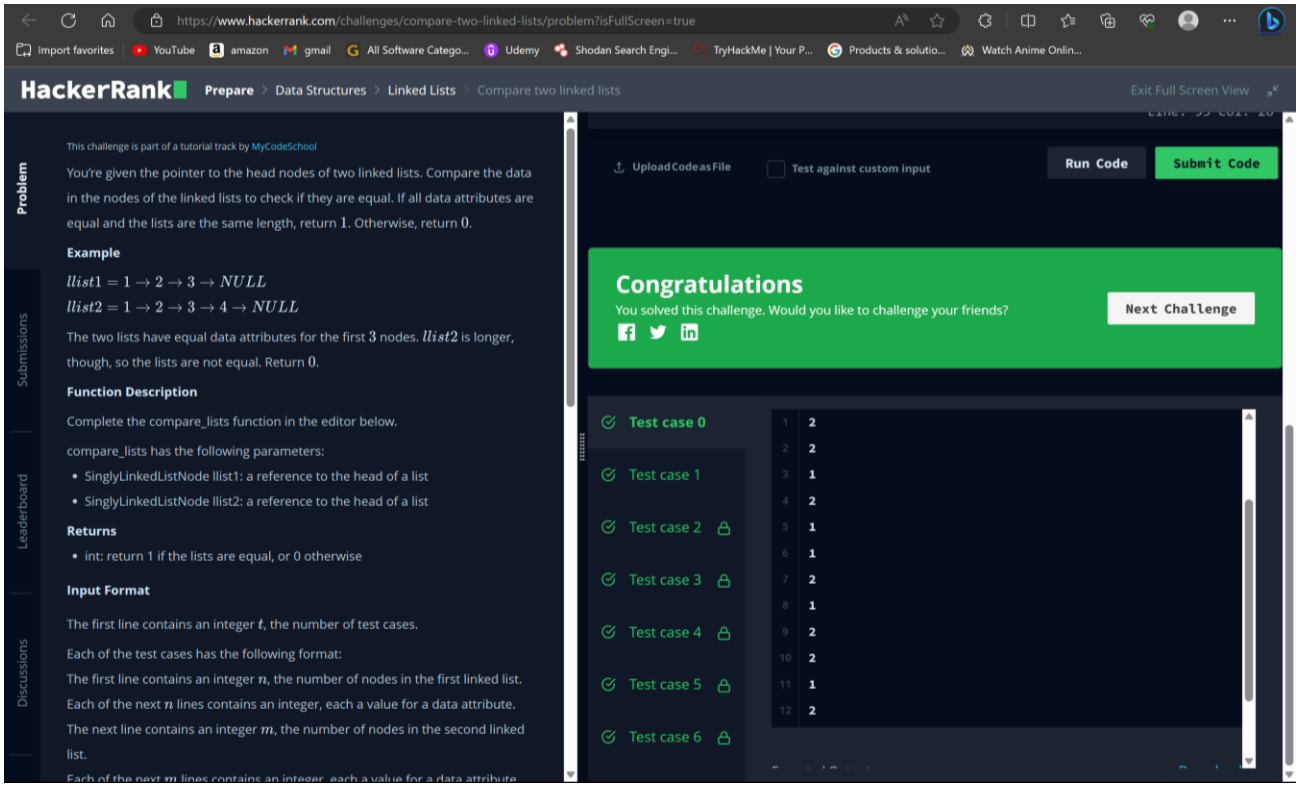
```
tests = int(input())

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()
```

5. Output:

Program 1:



Program 2:



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.