Course Name- AP-I Course Code- 21CSP-314

Experiment-1.3

Aim- Linked List: Demonstrate the concept of a Linked List

Objectives- Today, we will learn about the linked list data structure.

Problem1: https://www.hackerrank.com/challenges/compare-two-linked-lists/problem?isFullScreen=true

Problem2: https://www.hackerrank.com/challenges/insert-a-node-into-a-sorted-doubly-linked-list/problem?isFullScreen=true

Description-

Linked List can be defined as a collection of objects called nodes randomly stored in the memory. A node contains two fields i.e., data stored at that particular address and the pointer which contains the address of the next node in the memory. The last node of the list contains a pointer to the null. The list is not required to be continuously present in the memory. The node can reside anywhere in the memory and be linked together to make a list. This achieves optimized utilization of space. A linked list is a linear data structure with a series of connected nodes. A linked list can be defined as the randomly stored nodes in the memory. A node in the linked list contains two parts, i.e., the first is the data part and the second is the address part. The last node of the list contains a pointer to the null. After the array, a linked list is the second most used data structure. In a linked list, every link contains a connection to another link.

Types of Linked Lists

- Singly Linked list
- Doubly Linked list
- Circular Linked list
- Doubly Circular Linked list

Course Name- AP-I Course Code- 21CSP-314

Problem 1: 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.

Problem 2: Given a reference to the head of a doubly-linked list and an integer,data , create a new *DoublyLinkedListNode* object having data value data and insert it at the proper location to maintain the sort.

Code:

1.

Change Theme Language C++14

```
#include <bits/stdc++.h>
bool compare_lists(SinglyLinkedListNode* head1, SinglyLinkedListNode* head2) {
while (head1 && head2) {
        if (head1->data != head2->data) {
            return false;  // Elements are not equal
        }
        head1 = head1->next;
        head2 = head2->next;
    }

    // If both lists have reached the end, they are equal
    return head1 == nullptr && head2 == nullptr;
}


int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    int tests;
    cin >> tests;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    for (int tests_itr = 0; tests_itr < tests; tests_itr++) {
        SinglyLinkedList* llist1 = new SinglyLinkedList();

        int llist1_count;
        cin >> llist1_count;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

```

Course Name- AP-I Course Code- 21CSP-314

```
            int llist1_item;
            cin >> llist1_item;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            llist1->insert_node(llist1_item);
        }

        SinglyLinkedList* llist2 = new SinglyLinkedList();

        int llist2_count;
        cin >> llist2_count;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        for (int i = 0; i < llist2_count; i++) {
            int llist2_item;
            cin >> llist2_item;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            llist2->insert_node(llist2_item);
        }

        bool result = compare_lists(llist1->head, llist2->head);

        fout << result << "\n";
    }

    fout.close();

    return 0;
}
```

Line: 59 Col: 1

2.

Change Theme Language C++14

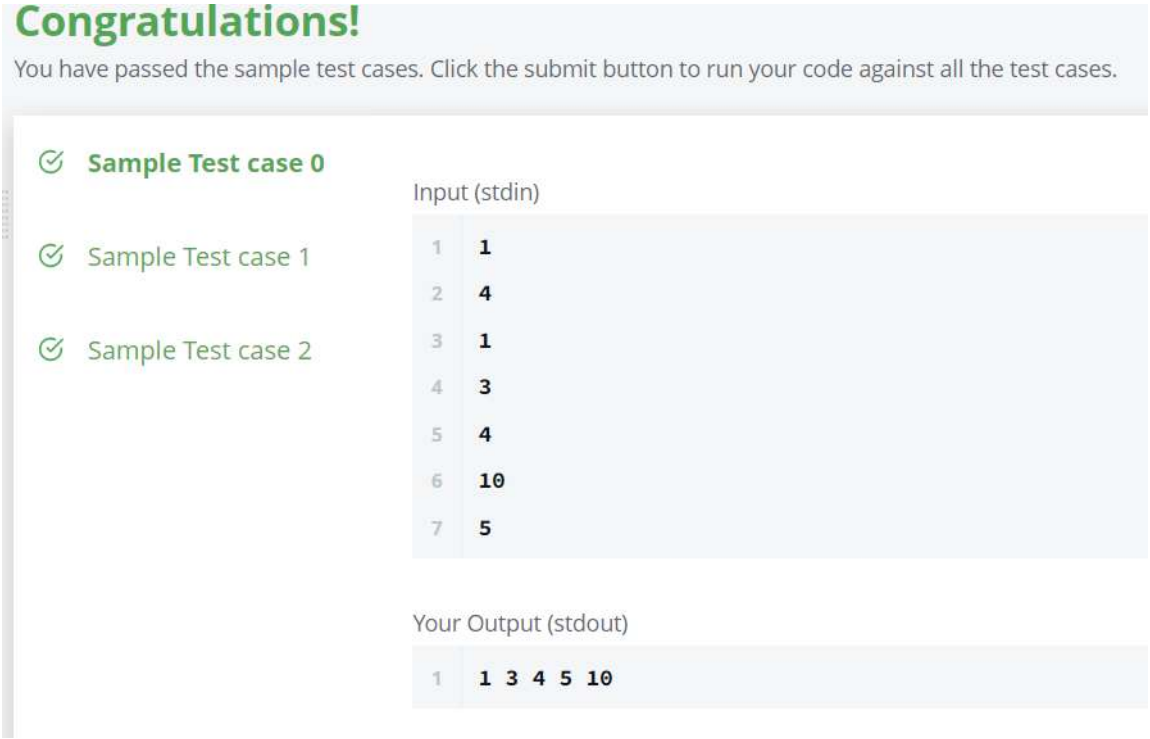
```
#include <bits/stdc++.h>

DoublyLinkedListNode* sortedInsert(DoublyLinkedListNode* llist, int data) {
    DoublyLinkedListNode* newNode = new DoublyLinkedListNode(data);

    if (!llist) {
        return newNode;
    }

    if (data < llist->data) {
        newNode->next = llist;
        llist->prev = newNode;
        return newNode;
    }

    DoublyLinkedListNode* current = llist;

    while (current->next && current->next->data < data) {
        current = current->next;
    }

    if (current->next) {
        current->next->prev = newNode;
    }

    newNode->next = current->next;
    newNode->prev = current;
    current->next = newNode;

    return llist;
```

Course Name- AP-I Course Code- 21CSP-314

Outcome-

Problem 1 outcome

Problem 2 Outcome



Course Name- AP-I Course Code- 21CSP-314

Learning Outcomes-

1. Learnt about the linked list data structure.
2. Learnt different types of linked list.
3. Learnt how to implement linked list.