



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.



## Experiment Title 1.3

**Student : ABHAY KUMAR**

**Branch: CSE**

**Semester: 5th**

**Subject :- Competitive Coding**

**UID: 20BCS9222**

**Section/Group:- 616-A**

**D.O.P:29/08/2022**

### Aim/Overview of the practical: LINKED-LIST

**Q1. 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.**

#### Example

head refers to the list 1 <-> 2 <-> 3 <-> 4 -> NULL

data = 3

Return a reference to the new list:

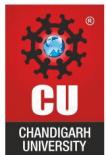
1 <-> 2 <-> 3 <-> 4 -> NULL

#### Function Description

Complete the sortedInsert function in the editor below.

sortedInsert has two parameters:

DoublyLinkedListNode pointer head: a reference to the head of a doubly-linked list



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.



int data: An integer denoting the value of the `data` field for the `DoublyLinkedListNode` you must insert into the list.

## Returns-

`DoublyLinkedListNode` pointer: a reference to the head of the list

Note: Recall that an empty list (i.e., where `head=NULL`) and a list with one element are sorted lists.

## Input Format-

The first line contains an integer `t`, the number of test cases.

Each of the test case is in the following format:

The first line contains an integer `n`, the number of elements in the linked list.

Each of the next `n` lines contains an integer, the data for each node of the linked list.

The last line contains an integer, `data`, which needs to be inserted into the sorted doubly-linked list.

## CODE :-

```
#include <bits/stdc++.h>
using namespace std;

class DoublyLinkedListNode {
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

```
public:  
    int data;  
  
    DoublyLinkedListNode *next;  
  
    DoublyLinkedListNode *prev;  
  
    DoublyLinkedListNode(int node_data) {  
  
        this->data = node_data;  
  
        this->next = nullptr;  
  
        this->prev = nullptr;  
  
    }  
  
};  
  
class DoublyLinkedList {  
  
public:  
    DoublyLinkedListNode *head;  
  
    DoublyLinkedListNode *tail;  
  
    DoublyLinkedList() {  
  
        this->head = nullptr;  
  
        this->tail = nullptr;  
  
    }  
  
    void insert_node(int node_data) {  
  
        DoublyLinkedListNode* node = new DoublyLinkedListNode(node_data);  
  
        if (!this->head) {  
  
            this->head = node;  
  
        }  
  
        else {  
  
            this->tail->next = node;  
  
            node->prev = this->tail;  
  
            this->tail = node;  
  
        }  
  
    }  
  
    void delete_node(int node_data) {  
  
        DoublyLinkedListNode* current = head;  
  
        while (current != nullptr) {  
  
            if (current->data == node_data) {  
  
                if (current == head) {  
  
                    head = current->next;  
  
                }  
  
                else if (current == tail) {  
  
                    tail = current->prev;  
  
                }  
  
                else {  
  
                    current->prev->next = current->next;  
  
                    current->next->prev = current->prev;  
  
                }  
  
                delete current;  
  
            }  
  
            current = current->next;  
  
        }  
  
    }  
  
    void display() {  
  
        DoublyLinkedListNode* current = head;  
  
        while (current != nullptr) {  
  
            cout << current->data << " ";  
  
            current = current->next;  
  
        }  
  
    }  
};
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

```
    } else {  
  
        this->tail->next = node;  
  
        node->prev = this->tail;  
  
    }  
  
    this->tail = node;  
  
}  
};  
  
void print_doubly_linked_list(DoublyLinkedListNode* node, string sep,  
ofstream& fout) {  
  
    while (node) {  
  
        fout << node->data;  
  
        node = node->next;  
  
        if (node) {  
  
            fout << sep;  
  
        }  
  
    }  
}  
  
void free_doubly_linked_list(DoublyLinkedListNode* node) {  
  
    while (node) {  
  
        DoublyLinkedListNode* temp = node;  
  
        node = node->next;  
  
        free(temp);  
    }  
}
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

```
}
```

```
}
```

```
DoublyLinkedListNode* sortedInsert(DoublyLinkedListNode* head, int data) {
```

```
    DoublyLinkedListNode* node = new DoublyLinkedListNode(data);  
    node->data = data;  
    node->next = node->prev = NULL;
```

```
    if(head==NULL)  
        return node;  
    if(head->data > data){  
        head->prev = node;  
        node->next = head;  
        return node;  
    }
```

```
    DoublyLinkedListNode* next = sortedInsert(head->next, data);  
    head->next = next;  
    next->prev = head;
```

```
    return head;
```

```
}
```

```
int main()  
{
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

```
//"SARTHAK GUPTA_20BCS4852"
ofstream fout(getenv("OUTPUT_PATH"));

int t;
cin >> t;

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

for (int t_itr = 0; t_itr < t; t_itr++) {

    DoublyLinkedList* llist = new DoublyLinkedList();

    int llist_count;
    cin >> llist_count;

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

    for (int i = 0; i < llist_count; i++) {

        int llist_item;
        cin >> llist_item;

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

        llist->insert_node(llist_item);

    }

    int data;
    cin >> data;

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

    DoublyLinkedListNode* llist1 = sortedInsert(llist->head, data);

    print_doubly_linked_list(llist1, " ", fout);

    fout << "\n";
}
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.



```
free_doubly_linked_list(llist1);
}
fout.close();
return 0;
```

## OUTPUT:-



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC GRADE A+  
ACCREDITED UNIVERSITY

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

**Example**

*head* refers to the list  $1 \leftrightarrow 2 \leftrightarrow 4 \rightarrow NULL$   
*data* = 3

Return a reference to the new list:  $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4 \rightarrow NULL$ .

**Function Description**

Complete the sortedInsert function in the editor below.

sortedInsert has two parameters:

- DoublyListNode pointer head: a reference to the head of a doubly-linked list
- int data: An integer denoting the value of the *data* field for the DoublyListNode you must insert into the list.

**Returns**

- DoublyListNode pointer: a reference to the head of the list

**Note:** Recall that an empty list (i.e. where *head* = *NULL*) and a list

Change Theme Language C++14

```
85 }
86 //SARTHAK GUPTA
87 //ID :-20BCS4852
88 v int main()
89 {
90     ofstream fout(getenv("OUTPUT_PATH"));
91
92     int t;
93     cin >> t;
94     cin.ignore(numeric_limits<streamsize>::max(), '\n');
95
96     for (int t_itr = 0; t_itr < t; t_itr++) {
97         DoublyLinkedList* llist = new DoublyLinkedList();
98
99         int llist_count;
100        cin >> llist_count;
101        cin.ignore(numeric_limits<streamsize>::max(), '\n');
102    }

```

Line: 87 Col: 3

Upload Code as File Test against custom input Run Code Submit Code

Problem

Submissions

Leaderboard



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC GRADE A+  
ACCREDITED UNIVERSITY

The screenshot shows a challenge titled "Inserting a Node Into a Sorted Doubly Linked List" on the HackerRank platform. On the left, there is a "Sample Input" section containing the following code:

```
STDIN Function
----- -----
1   t = 1
4   n = 4
1   node data values = 1, 3, 4, 10
3
4
10
5   data = 5
```

Below it is a "Sample Output" section showing the result:

```
1 3 4 5 10
```

On the right, a large green banner says "Congratulations!" followed by the message: "You have passed the sample test cases. Click the submit button to run your code against all the test cases." Below this, there are three "Sample Test case" entries, each with a green checkmark:

- Sample Test case 0: 1 1
- Sample Test case 1: 2 4
- Sample Test case 2: 3 1

Further down, there is a "Your Output (stdout)" section showing:

```
1 1 3 4 5 10
```

And an "Expected Output" section showing:

```
1 1 3 4 5 10
```

At the bottom right, there are download and market brief links, along with a timestamp: 14:26 02-09-2022.

**Q2. Given the pointer to the head node of a doubly linked list, reverse the order of the nodes in place. That is, change the next and prev pointers of the nodes so that the direction of the list is reversed. Return a reference to the head node of the reversed list.**

**Note:** The head node might be **NULL** to indicate that the list is empty.

**Function Description:-**

Complete the reverse function in the editor below.

reverse has the following parameter(s):

**DoublyLinkedListNode head: a reference to the head of a DoublyLinkedList**

**Returns:-**

- **DoublyLinkedListNode: a reference to the head of the reversed list**



## Input Format :-

The first line contains an integer  $t$ , the number of test cases.

Each test case is of the following format:

The first line contains an integer  $n$ , the number of elements in the linked list.

The next  $n$  lines contain an integer each denoting an element of the linked list

## CODE:-

```
#include <bits/stdc++.h>
using namespace std;

// SARTHAK GUPTA_ 20BCS4852
class DoublyLinkedListNode {
public:
    int data;
    DoublyLinkedListNode *next;
    DoublyLinkedListNode *prev;
    DoublyLinkedListNode(int node_data) {
        this->data = node_data;
        this->next = nullptr;
        this->prev = nullptr;
    }
};
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

```
class DoublyLinkedList {  
  
public:  
  
    DoublyLinkedListNode *head;  
  
    DoublyLinkedListNode *tail;  
  
    DoublyLinkedList() {  
  
        this->head = nullptr;  
  
        this->tail = nullptr;  
  
    }  
  
  
    void insert_node(int node_data) {  
  
        DoublyLinkedListNode* node = new DoublyLinkedListNode(node_data);  
  
        if (!this->head) {  
  
            this->head = node;  
  
        } else {  
  
            this->tail->next = node;  
  
            node->prev = this->tail;  
  
        }  
  
        this->tail = node;  
  
    }  
  
};  
  
void print_doubly_linked_list(DoublyLinkedListNode* node, string sep,  
ofstream& fout) {
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

```
while (node) {  
  
    fout << node->data;  
  
    node = node->next;  
  
  
    if (node) {  
  
        fout << sep;  
  
    }  
  
}  
  
void free_doubly_linked_list(DoublyLinkedListNode* node) {  
  
    while (node) {  
  
        DoublyLinkedListNode* temp = node;  
  
        node = node->next;  
  
        free(temp);  
  
    }  
  
}  
  
// Complete the reverse function below.  
  
/*  
 * For your reference:  
 * DoublyLinkedListNode {  
 *     int data;  
 *     DoublyLinkedListNode* next;  
 */
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

```
* DoublyLinkedListNode* prev;  
* };  
*/  
  
DoublyLinkedListNode* reverse(DoublyLinkedListNode* head)  
{  
    // Complete this function  
    // Do not write the main method.  
  
    DoublyLinkedListNode *current = head;  
  
    DoublyLinkedListNode *temp = NULL;  
  
    while ( current != NULL) {  
  
        temp = current -> prev;  
  
        current -> prev = current -> next;  
  
        current -> next = temp;  
  
        current = current -> prev;  
  
    }  
  
    if (temp != NULL)  
  
        head = temp -> prev;  
  
    return head;  
}  
  
int main(){  
  
    ofstream fout(getenv("OUTPUT_PATH"));  
  
    int t;
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

```
cin >> t;  
  
cin.ignore(numeric_limits<streamsize>::max(), '\n');  
  
for (int t_itr = 0; t_itr < t; t_itr++) {  
  
    DoublyLinkedList* llist = new DoublyLinkedList();  
  
    int llist_count;  
  
    cin >> llist_count;  
  
    cin.ignore(numeric_limits<streamsize>::max(), '\n');  
  
    for (int i = 0; i < llist_count; i++) {  
  
        int llist_item;  
  
        cin >> llist_item;  
  
        cin.ignore(numeric_limits<streamsize>::max(), '\n');  
  
        llist->insert_node(llist_item);  
  
    }  
  
    DoublyLinkedListNode* llist1 = reverse(llist->head);  
  
    print_doubly_linked_list(llist1, " ", fout);  
  
    fout << "\n";  
  
    free_doubly_linked_list(llist1);  
  
}  
  
fout.close();  
  
return 0;  
}
```



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

## Result/Output:

The screenshot shows a Windows desktop environment with a taskbar at the top. The taskbar has several icons, including a search bar, a mail icon, a file explorer icon, a browser icon, a calculator icon, a power icon, and a system tray icon. The main window is a browser displaying the HackerRank challenge "Reverse a doubly linked list". The URL in the address bar is <https://www.hackerrank.com/challenges/reverse-a-doubly-linked-list/problem?isFullScreen=true>. The challenge page shows the problem statement, sample input (1 4 1 2 3 4), sample output (4 3 2 1), and an explanation. On the right side of the page is a code editor with the following C++ code:

```
79 // 
80 DoublyLinkedListNode* reverse(DoublyLinkedListNode* llist) {
81     DoublyLinkedListNode* prev = NULL;
82     DoublyLinkedListNode* curr = llist;
83     DoublyLinkedListNode* next = curr->next;
84     while (curr != NULL) {
85         curr->next = prev;
86         prev = curr;
87         curr = next;
88         if (curr != NULL)
89             next = curr->next;
90     }
91     return prev;
92 }
93
94
95
96
97
98
99
100 }
```

The code editor interface includes tabs for "Change Theme" and "Language" (set to C++14), a status bar showing "Line: 85 Col: 19", and buttons for "Upload Code as File", "Test against custom input", "Run Code", and "Submit Code".



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC GRADE A+  
ACCREDITED UNIVERSITY

(12) WhatsApp Reverse a doubly linked list | Hack

https://www.hackerrank.com/challenges/reverse-a-doubly-linked-list/problem?isFullScreen=true

Gmail YouTube Maps GeeksforGeeks | A... Digital Electronics a... www.programiz.com lo-au.vlabs.ac.in/las... Blackboard Learn SHL simplesnippets

## HackerRank Prepare > Data Structures > Linked Lists > Reverse a doubly linked list

Exit Full Screen View

Leaderboard Discussions Editorial

Return a reference to the head of your reversed list. The provided code will print the reverse array as a one line of space-separated integers for each test case.

**Sample Input**

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

**Sample Output**

```
4 3 2 1
```

**Explanation**

The initial doubly linked list is:  $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4 \rightarrow \text{NULL}$

**Congratulations!**  
You have passed the sample test cases. Click the submit button to run your code against all the test cases.

**Sample Test case 0**

Input (stdin)	Download
1 2 3 4 5 6	

**Sample Test case 1**

Input (stdin)	Download
1 2 3 4 5 6	

**Sample Test case 2**

Input (stdin)	Download
1 2 3 4 5 6	

**Your Output (stdout)**

```
1 4 3 2 1
```

33°C Haze 10:35 08-09-2022

https://www.hackerrank.com/challenges/reverse-a-doubly-linked-list/problem?isFullScreen=true

Gmail YouTube Maps GeeksforGeeks | A... Digital Electronics a... www.programiz.com lo-au.vlabs.ac.in/las... Blackboard Learn SHL simplesnippets

## HackerRank Prepare > Data Structures > Linked Lists > Reverse a doubly linked list

Exit Full Screen View

Problem Submissions Leaderboard

The first line contains an integer  $t$ , the number of test cases. Each test case is of the following format:

- The first line contains an integer  $n$ , the number of elements in the linked list.
- The next  $n$  lines contain an integer each denoting an element of the linked list.

**Constraints**

- $1 \leq t \leq 10$
- $0 \leq n \leq 1000$
- $0 \leq \text{DoublyLinkedListNode}.data \leq 1000$

**Output Format**

Return a reference to the head of your reversed list. The provided code will print the reverse array as a one line of space-separated integers for each test case.

**Sample Input**

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

**Expected Output**

```
1 4 3 2 1
```

33°C Haze 10:43 08-09-2022

## Learning Outcomes:



# DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC  
GRADE A+  
ACCREDITED UNIVERSITY

- 
- 1) Learn about C++.
  - 2) Learn about LinkedList and how to implement it
  - 3) Learn About DoublyLinkedList.