Rajalakshmi Engineering College

Name: Tejeshwaran P
Email: 241801292@rajalakshmi.edu.in
Roll no: 241801292
Phone: 6383048813
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Sam is learning about two-way linked lists. He came across a problem where he had to populate a two-way linked list and print the original as well as the reverse order of the list. Assist him with a suitable program.

Input Format

The first line of input consists of an integer n, representing the number of elements in the list.

The second line consists of n space-separated integers, representing the elements.

Output Format

The first line displays the message: "List in original order:"

The second line displays the elements of the doubly linked list in the original order.

The third line displays the message: "List in reverse order:"

The fourth line displays the elements of the doubly linked list in reverse order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
Output: List in original order:
1 2 3 4 5
List in reverse order:
5 4 3 2 1

Answer

```
// You are using GCC #include <iostream> using
namespace std;

// Node class representing each element in the                doubly
linked list class Node { public:    int data;   // Store        the data of
the node     Node* next; // Pointer to the next               node
   Node* prev; // Pointer to the previous node

   // Constructor to initialize the node with data
   Node(int value) {        data =
value;        next = nullptr;        prev
= nullptr;
   }
};

// DoublyLinkedList                  class to                manage the
list of nodes class
DoublyLinkedList {
```

```
public:
   Node* head; // Pointer to the head of the list
Node* tail; // Pointer to the tail of the list

   // Constructor to initialize the list as empty
   DoublyLinkedList() {        head = nullptr;
      tail = nullptr;
}

   // Method to insert a node at the end of the list
   void insertEnd(int value) {
      Node* newNode = new Node(value); // Create a new node        if (head ==
nullptr) {
         head = tail = newNode; // If the list is empty, both head and tail point to
the new node
      } else {
         tail->next = newNode;  // Link the new node to the last node's next
newNode->prev = tail;  // Set the new node's previous pointer to the last node
         tail = newNode;       // Update the tail to the new node
      }
   }

   // Method to print the list in original order (head                        to tail)
void printOriginalOrder() {        Node* current =                            head;
while (current != nullptr) {          cout << current-                        >data << " ";
         current = current->next;
      }
   }

   // Method to print the list in reverse order (tail                         to head)
   void printReverseOrder() {        Node* current =                          tail;
while (current != nullptr) {          cout << current-                        >data << " ";
current = current->prev;
      }
   }
};
int main() {
   int n;
   cin >> n;  // Read the number of elements

   DoublyLinkedList list; // Create an empty doubly
linked list

   // Read the n space-separated integers and
insert them into the list     for (int i = 0; i < n; ++i) {
int value;        cin >> value;
```

```
        list.insertEnd(value); // Insert each element at                              the end of
the list
    }

    // Output the list in original order     cout << "List
in original order: ";     list.printOriginalOrder();
cout << endl;

    // Output the list in reverse order     cout << "List in reverse
order: ";     list.printReverseOrder();     cout << endl;
```

Status : Correct*Marks*   *: 10/10*