# Rajalakshmi Engineering College

Name: Shremathi K
Email: 240701504@rajalakshmi.edu.in
Roll no:
Phone: 8870649491
Branch: REC
Department: I CSE FE
Batch: 2028
Degree: B.E - CSE

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

Krishna needs to create a doubly linked list to store and display a sequence of integers. Your task is to help write a program to read a list of integers from input, store them in a doubly linked list, and then display the list.

*Input Format*

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

The second line of input consists of n space-separated integers.

*Output Format*

The output prints a single line displaying the integers in the order they were added to the doubly linked list, separated by spaces.

If nothing is added (i.e., the list is empty), it will display "List is empty".

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 5
1 2 3 4 5

Output: 1 2 3 4 5

***Answer***

```
#include <iostream>
using namespace std;

// Structure for a doubly linked list node
struct Node {
   int data;
   Node* prev;
   Node* next;
};

// Function to insert a node at the end
void insertAtEnd(Node*& head, Node*& tail, int value) {
   Node* newNode = new Node;
   newNode->data = value;
   newNode->prev = nullptr;
   newNode->next = nullptr;

   if (head == nullptr) {
      head = tail = newNode;
   } else {
      tail->next = newNode;
      newNode->prev = tail;
      tail = newNode;
   }
}

// Function to display the list
void displayList(Node* head) {
    if (head == nullptr) {
        cout << "List is empty" << endl;
        return;
    }

    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

// Main function
int main() {
    int n;
    cin >> n;

    Node* head = nullptr;
    Node* tail = nullptr;

    for (int i = 0; i < n; ++i) {
        int val;
        cin >> val;
        insertAtEnd(head, tail, val);
    }

    displayList(head);

    return 0;
}
```

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

2. Problem Statement

Imagine you're managing a store's inventory list, and some products were accidentally entered multiple times. You need to remove the duplicate products from the list to ensure each product appears only once.

You have an unsorted doubly linked list of product IDs. Some of these product IDs may appear more than once, and your goal is to remove any duplicates.

***Input Format***

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

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

***Output Format***

The output prints the final after removing duplicate nodes, separated by a space.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 10
12 12 10 4 8 4 6 4 4 8

Output: 8 4 6 10 12

***Answer***

```
#include <iostream>
#include <unordered_set>
using namespace std;

// Structure of a doubly linked list node
struct Node {
   int data;
   Node* prev;
   Node* next;
};

// Insert at front (to build list in reverse)
void insertAtFront(Node*& head, Node*& tail, int value) {
   Node* newNode = new Node;
   newNode->data = value;
   newNode->prev = nullptr;
```

```
    newNode->next = head;

    if (head != nullptr) {
        head->prev = newNode;
    } else {
        tail = newNode;  // If list was empty
    }

    head = newNode;
}

// Remove duplicates from the list
void removeDuplicates(Node*& head, Node*& tail) {
    unordered_set<int> seen;
    Node* current = head;

    while (current != nullptr) {
        if (seen.find(current->data) != seen.end()) {
            // Duplicate found - remove the node
            Node* toDelete = current;
            if (toDelete->prev) toDelete->prev->next = toDelete->next;
            if (toDelete->next) toDelete->next->prev = toDelete->prev;

            if (toDelete == head) head = toDelete->next;
            if (toDelete == tail) tail = toDelete->prev;

            current = current->next;
            delete toDelete;
        } else {
            seen.insert(current->data);
            current = current->next;
        }
    }
}

// Display the list
void displayList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    int n;
    cin >> n;

    Node* head = nullptr;
    Node* tail = nullptr;

    // Read input and insert in front (so last in input appears first)
    for (int i = 0; i < n; ++i) {
        int value;
        cin >> value;
        insertAtFront(head, tail, value);
    }

    removeDuplicates(head, tail);
    displayList(head);

    return 0;
}
```

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

3. Problem Statement

You are required to implement a program that deals with a doubly linked list.

The program should allow users to perform the following operations:

Insertion at the End: Insert a node with a given integer data at the end of the doubly linked list.Insertion at a given Position: Insert a node with a given integer data at a specified position within the doubly linked list.Display the List: Display the elements of the doubly linked list.

***Input Format***

The first line of input consists of an integer n, representing the number of elements to be initially inserted into the doubly linked list.

The second line consists of n space-separated integers, denoting the elements to be inserted at the end.

The third line consists of integer m, representing the new element to be inserted.

The fourth line consists of an integer p, representing the position at which the new element should be inserted (1-based indexing).

*Output Format*

If p is valid, display the elements of the doubly linked list after performing the insertion at the specified position.

If p is invalid, display "Invalid position" in the first line and the second line prints the original list.

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: 5
10 25 34 48 57
35
4

Output: 10 25 34 35 48 57

*Answer*

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

// Structure for a node
struct Node {
   int data;
   Node* prev;
   Node* next;
};

// Insert node at the end of the list
void insertAtEnd(Node*& head, Node*& tail, int value) {
   Node* newNode = new Node{value, nullptr, nullptr};
```

```
    if (head == nullptr) {
        head = tail = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        tail = newNode;
    }
}

// Insert node at a given position (1-based indexing)
bool insertAtPosition(Node*& head, Node*& tail, int value, int pos) {
    if (pos < 1) return false;

    Node* newNode = new Node{value, nullptr, nullptr};

    if (pos == 1) {
        newNode->next = head;
        if (head != nullptr) head->prev = newNode;
        else tail = newNode; // If list was empty
        head = newNode;
        return true;
    }

    Node* current = head;
    int count = 1;

    while (current != nullptr && count < pos - 1) {
        current = current->next;
        count++;
    }

    if (current == nullptr) return false;

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

    if (current->next != nullptr) {
        current->next->prev = newNode;
    } else {
        tail = newNode; // Inserting at the end
    }
    current->next = newNode;
    return true;
}

// Display the list
void displayList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    int n;
    cin >> n;

    Node* head = nullptr;
    Node* tail = nullptr;

    // Insert initial elements at end
    for (int i = 0; i < n; ++i) {
        int val;
        cin >> val;
        insertAtEnd(head, tail, val);
    }

    int m, p;
    cin >> m >> p;

    bool inserted = insertAtPosition(head, tail, m, p);

    if (!inserted) {
        cout << "Invalid position" << endl;
    }

    displayList(head);

    return 0;
}
```

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