//Practical 2C-Write a program to create Doubly Linked List and sort the elements in the linked list

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

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

// Function to create a new node
Node* createNode(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

// Function to insert node at the end
void insertEnd(Node*& head, int value) {
    Node* newNode = createNode(value);
    if (head == NULL) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
```

```
        newNode->prev = temp;
    }
}

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

    cout << "Doubly Linked List: ";
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

// Function to sort the doubly linked list using Bubble Sort
void sortList(Node* head) {
    if (head == NULL) return;

    bool swapped;
    Node* ptr;
    Node* last = NULL;

    do {
        swapped = false;
        ptr = head;
```

```
        while (ptr->next != last) {
            if (ptr->data > ptr->next->data) {
                // Swap data
                int temp = ptr->data;
                ptr->data = ptr->next->data;
                ptr->next->data = temp;
                swapped = true;
            }
            ptr = ptr->next;
        }
        last = ptr;
    } while (swapped);
}

int main() {
    Node* head = NULL;
    int n, value, i;

    cout << "Enter the number of nodes: ";
    cin >> n;

    for (i = 0; i < n; i++) {
        cout << "Enter value for node " << i + 1 << ": ";
        cin >> value;
        insertEnd(head, value);
    }

    cout << "\nOriginal List:\n";
    displayList(head);
```

```
    sortList(head);

    cout << "\nSorted List:\n";
    displayList(head);

    return 0;
}
```

Practical 3A-Implement the concept of Stack with Push,Pop,Display and Exit operations

```
#include <iostream>
#define MAX 5
using namespace std;

int stack[MAX];
int top = -1;

void push();
void pop();
void display();

int main() {
    int choice;

    while (true) {
        cout << "\nStack Operations Menu:\n";
        cout << "1. Push\n2. Pop\n3. Display\n4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
```

```
        switch (choice) {
            case 1: push(); break;
            case 2: pop(); break;
            case 3: display(); break;
            case 4: cout << "Exiting program...\n"; return 0;
            default: cout << "Invalid choice! Please try again.\n";
        }
    }

    return 0;
}

// Push function
void push() {
    if (top == MAX - 1) {
        cout << "Stack is full! (Overflow)\n";
    } else {
        int val;
        cout << "Enter element to push: ";
        cin >> val;
        top++;
        stack[top] = val;
        cout << val << " pushed onto the stack.\n";
    }
}

// Pop function
void pop() {
    if (top == -1) {
```

```
        cout << "Stack is empty! (Underflow)\n";
    } else {
        cout << "Popped element: " << stack[top] << endl;
        top--;
    }
}

// Display function
void display() {
    if (top == -1) {
        cout << "Stack is empty!\n";
    } else {
        cout << "Stack elements (top to bottom):\n";
        for (int i = top; i >= 0; i--) {
            cout << stack[i] << endl;
        }
    }
}
```