Naafiul Hossain
ESE224
115107623
Tuesday 10-12:50am

Problem 1:

Main.cpp

```cpp
//Naafiul Hossain
//SBU ID: 115107623

#include <iostream>
using namespace std;

template<typename Type>
struct Node {
    Type value;
    Node* next;
    Node() : next(nullptr) {}
    Node(Type x, Node* next = nullptr) : value(x), next(next) {}
    ~Node() {}
};

template<typename Type>
Node<Type>* createLinkedList(Type* arr, int n) {
    if (n <= 0) {
        cout << "Please check your input!" << endl;
        return nullptr;
    }
    Node<Type>* head = new Node<Type>(arr[0]);
    Node<Type>* tmp = head;
    for (int i = 1; i < n; i++) {
        tmp->next = new Node<Type>(arr[i]);
        tmp = tmp->next;
    }
    return head;
}
```

```cpp
template <typename Type>
void printLinkedList(Node<Type>* head) {
    while (head) {
        cout << head->value << " ";
        head = head->next;
    }
    cout << endl;
}

template <typename Type>
Node<Type>* reverse(Node<Type>* head) {
    Node<Type>* current = head;
    Node<Type>* pre = nullptr;
    Node<Type>* next = nullptr;

    while (current) {
        next = current->next;
        current->next = pre;
        pre = current;
        current = next;
    }
    return pre;
}
```

```cpp
int main() {
    int arr[] = { 1, 2, 3, 4, 5, 6, 7 };
    cout << "Create linked list from array: " << endl;
    Node<int>* node = createLinkedList(arr, sizeof(arr) / sizeof(arr[0]));
    printLinkedList(node);
    cout << "Reverse linked list: " << endl;
    Node<int>* reversed_node = reverse(node);
    printLinkedList(reversed_node);
    return 0;
}
```

Screenshot of the running program:

```
Create linked list from array:
1 2 3 4 5 6 7
Reverse linked list:
7 6 5 4 3 2 1

C:\Users\Naafiul Hossain\Documents\
```

Problem 2

main.h

```
//Naafiul Hossain
//SBU ID: 115107623
#include <iostream>
using namespace std;

bool isSorted(int* arr, int size) {
    for (int i = 1; i < size; i++) {
        if (arr[i - 1] > arr[i]) {
            return false;
        }
    }
    return true;
}

void bubblesort(int* arr, int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - 1 - i; j++) {
            if (arr[j] > arr[j + 1]) {
                // Swap the elements if they are out of order
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
```

```
int main() {
    int n;
    cout << "Enter the number of elements in the array: ";
    cin >> n;

    int* arr = new int[n];

    cout << "Enter the values for the array:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    bool sorted = isSorted(arr, n);
    cout << "isSorted: " << (sorted ? "true" : "false") << endl;

    if (!sorted) {
        cout << "Sorting the array using bubblesort..." << endl;
        bubblesort(arr, n);
    }

    cout << "Sorted array: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    delete[] arr;
    return 0;
}
```

Running of the Program:

```
Enter the number of elements in the array: 5
Enter the values for the array:
1 3 4 2 6
isSorted: false
Sorting the array using bubblesort...
Sorted array: 1 2 3 4 6
```

Problem 3

Main.cpp

```cpp
//Naafiul Hossain
//SBU ID: 115107623

#include <iostream>
using namespace std;

template<typename Type>
struct Node {
    Type value;
    Node* next;
    Node() : next(nullptr) {}
    Node(Type x, Node* next = nullptr) : value(x), next(next) {}
    ~Node() {}
};

template<typename Type>
Node<Type>* createLinkedList(Type* arr, int n) {
    if (n <= 0) {
        cout << "Please check your input!" << endl;
        return nullptr;
    }
    Node<Type>* head = new Node<Type>(arr[0]);
    Node<Type>* tmp = head;
    for (int i = 1; i < n; i++) {
        tmp->next = new Node<Type>(arr[i]);
        tmp = tmp->next;
    }
    return head;
}
```

```cpp
template <typename Type>
void printLinkedList(Node<Type>* head) {
    while (head) {
        cout << head->value << " ";
        head = head->next;
    }
    cout << endl;
}

template <typename Type>
Node<Type>* reverse(Node<Type>* head) {
    Node<Type>* current = head;
    Node<Type>* pre = nullptr;
    Node<Type>* next = nullptr;

    while (current) {
        next = current->next;
        current->next = pre;
        pre = current;
        current = next;
    }
    return pre;
}
```

```cpp
template <typename Type>
void removeDuplicates(Node<Type>* head) {
    if (!head) return;  // Handle empty list

    Node<Type>* current = head;

    while (current && current->next) {
        if (current->value == current->next->value) {
            Node<Type>* duplicate = current->next;
            current->next = duplicate->next;
            delete duplicate;
        }
        else {
            current = current->next;
        }
    }
}

int main() {
    int arr[] = { 0,1,3,3,4,4,5,6,6,6,6,6,6,7 };
    cout << "Create linked list from array: " << endl;
    Node<int>* node = createLinkedList(arr, sizeof(arr) / sizeof(arr[0]));
    printLinkedList(node);

    cout << "Removing duplicates: " << endl;
    removeDuplicates(node);
    printLinkedList(node);

    return 0;
}
```

Screenshot of the running program:

```
Microsoft Visual Studio Debu

Create linked list from array:
0 1 3 3 4 4 5 6 6 6 6 6 6 7
Removing duplicates:
0 1 3 4 5 6 7
```

Problem 4

Main.cpp

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

template <typename T>
class DetailedLinkedList {
private:
    struct Node {
        T data;
        Node* next;
        Node(T value) : data(value), next(nullptr) {}
    };

    Node* head;
    size_t count;

public:
    DetailedLinkedList() : head(nullptr), count(0) {}

    void push_back(T value) {
        Node* newNode = new Node(value);
        if (!head) {
            head = newNode;
        }
        else {
            Node* current = head;
            while (current->next) {
                current = current->next;
            }
            current->next = newNode;
```

```cpp
bool deleteNode(T value) {
    if (!head) {
        return false;  // List is empty
    }

    if (head->data == value) {
        Node* temp = head;
        head = head->next;
        delete temp;
        count--;
        return true;
    }

    Node* current = head;
    while (current->next && current->next->data != value) {
        current = current->next;
    }

    if (current->next) {
        Node* temp = current->next;
        current->next = current->next->next;
        delete temp;
        count--;
        return true;
    }

    return false;  // Node not found
}
```

```cpp
    void displayDeletedNodeDetails(T value) {
        if (deleteNode(value)) {
            cout << "Deleted node with value " << value << " at position " << count + 1 << endl;
        }
        else {
            cout << "Node with value " << value << " not found in the list." << endl;
        }
    }

    size_t size() const {
        return count;
    }

    void displayList() {
        Node* current = head;
        while (current) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }

    ~DetailedLinkedList() {
        while (head) {
            Node* temp = head;
            head = head->next;
            delete temp;
        }
    }
};
```

```cpp
int main() {
    DetailedLinkedList<int> list;

    list.push_back(1);
    list.push_back(2);
    list.push_back(3);
    list.push_back(4);

    cout << "Original List: ";
    list.displayList();

    list.displayDeletedNodeDetails(3);  // Delete node with value 3
    cout << "Updated List: ";
    list.displayList();

    list.displayDeletedNodeDetails(5);  // Attempt to delete a non-existent node
    cout << "Updated List: ";
    list.displayList();

    return 0;
}
```

Screenshot of the Program running:

```
Microsoft Visual Studio Debu

Original List: 1 2 3 4
Deleted node with value 3 at position 4
Updated List: 1 2 4
Node with value 5 not found in the list
Updated List: 1 2 4
```

Problem 5

Main.cpp

```cpp
//Naafiul Hossain
//SBU ID: 115107623

#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
using namespace std;

struct Node {
    int data;
    Node* next;
    Node(int value) : data(value), next(nullptr) {}
};

class LinkedList {
public:
    Node* head;
    LinkedList() : head(nullptr) {}

    void push_back(int value) {
        Node* newNode = new Node(value);
        if (!head) {
            head = newNode;
        }
        else {
            Node* current = head;
            while (current->next) {
                current = current->next;
            }
            current->next = newNode;
        }
```

```
public:
    Node* head;
    LinkedList() : head(nullptr) {}

    void push_back(int value) {
        Node* newNode = new Node(value);
        if (!head) {
            head = newNode;
        }
        else {
            Node* current = head;
            while (current->next) {
                current = current->next;
            }
            current->next = newNode;
        }
    }

    void display() {
        Node* current = head;
        while (current) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
};
```

```
int main() {
    fstream inputFile("input.txt");
    string line;

    // Create two linked lists for the values in input.txt
    LinkedList list1, list2;
    vector<int> values1, values2;

    // Read the first line of values
    if (getline(inputFile, line)) {
        stringstream ss(line);
        int value;
        while (ss >> value) {
            values1.push_back(value);
        }
    }

    // Read the second line of values
    if (getline(inputFile, line)) {
        stringstream ss(line);
        int value;
        while (ss >> value) {
            values2.push_back(value);
        }
    }

    // Populate list1 and list2 with the extracted values
    for (int value : values1) {
        list1.push_back(value);
    }
```

```
    // Manually delete all three lists
    current1 = list1.head;
    while (current1) {
        Node* temp = current1;
        current1 = current1->next;
        delete temp;
    }

    current2 = list2.head;
    while (current2) {
        Node* temp = current2;
        current2 = current2->next;
        delete temp;
    }

    current3 = list3.head;
    while (current3) {
        Node* temp = current3;
        current3 = current3->next;
        delete temp;
    }

    inputFile.close();
    return 0;
}
```

Screenshot of the running program:

```
Microsoft Visual Studio Debu

List 1: 1 2 3 4
List 2: 10 20 30 40
List 3: 11 22 33 44
```

Problem 6

Main.cpp

```cpp
//Naafiul Hossain
//115107623
#include <iostream>
using namespace std;

template<typename Type>
struct Node {
    Type value;
    Node* next;
    Node() : next(nullptr) {}
    Node(Type x, Node* next = nullptr) : value(x), next(next) {}
    ~Node() {}
};

template<typename Type>
Node<Type>* createLinkedList(Type* arr, int n) {
    if (n <= 0) {
        cout << "Please check your input!" << endl;
        return nullptr;
    }
    Node<Type>* head = new Node<Type>(arr[0]);
    Node<Type>* tmp = head;
    for (int i = 1; i < n; i++) {
        tmp->next = new Node<Type>(arr[i]);
        tmp = tmp->next;
    }
```

put

```cpp
template<typename Type>
Node<Type>* createLinkedList(Type* arr, int n) {
    if (n <= 0) {
        cout << "Please check your input!" << endl;
        return nullptr;
    }
    Node<Type>* head = new Node<Type>(arr[0]);
    Node<Type>* tmp = head;
    for (int i = 1; i < n; i++) {
        tmp->next = new Node<Type>(arr[i]);
        tmp = tmp->next;
    }
    return head;
}

template <typename Type>
void printLinkedList(Node<Type>* head) {
    while (head) {
        cout << head->value << " ";
        head = head->next;
    }
    cout << endl;
}
```

```cpp
int main() {
    int arr1[] = { 1, 3, 5,7 };
    int arr2[] = { 2, 4, 6,8 };
    int arr3[] = { 0, 9, 10,11 };

    Node<int>* list1 = createLinkedList(arr1, sizeof(arr1) / sizeof(arr1[0]));
    Node<int>* list2 = createLinkedList(arr2, sizeof(arr2) / sizeof(arr2[0]));
    Node<int>* list3 = createLinkedList(arr3, sizeof(arr3) / sizeof(arr3[0]));

    cout << "Merged linked list: " << endl;
    Node<int>* mergedList = mergeLinkedLists(list1, list2, list3);
    printLinkedList(mergedList);

    return 0;
}
```

Running solution:

```
Merged linked list:
1 3 5 7 2 4 6 8 0 9 10 11
```