Naafiul Hossain
ESE224
115107623
Tuesday 10-12:50am

Problem 1:

Main.cpp

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

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

class CEAS {
public:
    virtual void displayInfo();
    virtual void requirements();
    virtual void PrintCores();
};

class ECE : public CEAS {
public:
    void displayInfo() override;
    void PrintCores() override;
};

class ESE : public CEAS {
public:
    void displayInfo() override;
    void requirements() override;
    void PrintCores() override;
};

void CEAS::displayInfo() {
    cout << "CEAS students need..." << endl;
}

void CEAS::requirements() {
    cout << "Req1: Completion of at least 120 credit hours of passing work" << endl;
    cout << "Req2: A minimum cumulative grade point average of 2.00" << endl;
```

```
void CEAS::requirements() {
    cout << "Req1: Completion of at least 120 credit hours of passing work" << endl;
    cout << "Req2: A minimum cumulative grade point average of 2.00" << endl;
}

void CEAS::PrintCores() {}

void ECE::displayInfo() {
    cout << "ECE students need..." << endl;
}

void ECE::PrintCores() {
    cout << "ECE core1: Computer Arch" << endl;
    cout << "ECE core2: RealTimeOS" << endl;
    cout << "ECE core3: VHDL" << endl;
}

void ESE::displayInfo() {
    cout << "ESE students need..." << endl;
}



void ESE::requirements() {
    CEAS::requirements();  // Call the base class requirements
    cout << "ESE-specific requirement..." << endl;
}
```

```
void ESE::PrintCores() {
    cout << "ESE core1: Electromagnetic and Transmission Line Theroy" << endl;
    cout << "ESE core2:  Control Theroy" << endl;
}

void displayInfo(CEAS* p) {
    p->displayInfo();
}

void printRequirements(CEAS* p) {
    p->requirements();
}

void printCores(CEAS* p) {
    p->PrintCores();
}

int main() {
    CEAS* ceasStudent = new CEAS;
    displayInfo(ceasStudent);
    printRequirements(ceasStudent);
    printCores(ceasStudent);
    cout << "................" << endl;

    CEAS* eseStudent = new ESE;
    displayInfo(eseStudent);
    printRequirements(eseStudent);
    printCores(eseStudent);
    cout << "................." << endl;

    CEAS* eceStudent = new ECE;
    displayInfo(eceStudent);
    printRequirements(eceStudent);
    printCores(eceStudent);
```

```
int main() {
    CEAS* ceasStudent = new CEAS;
    displayInfo(ceasStudent);
    printRequirements(ceasStudent);
    printCores(ceasStudent);
    cout << "................" << endl;

    CEAS* eseStudent = new ESE;
    displayInfo(eseStudent);
    printRequirements(eseStudent);
    printCores(eseStudent);
    cout << "................" << endl;

    CEAS* eceStudent = new ECE;
    displayInfo(eceStudent);
    printRequirements(eceStudent);
    printCores(eceStudent);

    // Don't forget to delete the allocated memory
    delete ceasStudent;
    delete eseStudent;
    delete eceStudent;

    return 0;
}
```

Screenshot of the running program:

```
Microsoft Visual Studio Debu
CEAS students need...
Req1: Completion of at least 120 credit hours of passing work
Req2: A minimum cumulative grade point average of 2.00
................
ESE students need...
Req1: Completion of at least 120 credit hours of passing work
Req2: A minimum cumulative grade point average of 2.00
ESE-specific requirement...
ESE core1: Electromagnetic and Transmission Line Theroy
ESE core2:  Control Theroy
.................
ECE students need...
Req1: Completion of at least 120 credit hours of passing work
Req2: A minimum cumulative grade point average of 2.00
ECE core1: Computer Arch
ECE core2: RealTimeOS
ECE core3: VHDL
```

Problem 2

main.h

```cpp
#include <iostream>

// Node class for doubly linked list
class Node {
public:
    int data;
    Node* next;
    Node* prev;

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

// Doubly linked list class
class DoublyLinkedList {
public:
    Node* head;

    DoublyLinkedList() : head(nullptr) {}

    // Function to insert a node at the end of the list
    void insert(int value) {
        Node* newNode = new Node(value);

        if (!head) {
            head = newNode;
        }
        else {
            Node* temp = head;
            while (temp->next) {
                temp = temp->next;
```

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

    // Function to print the doubly linked list
    void print() {
        Node* temp = head;
        while (temp) {
            std::cout << temp->data << " ";
            temp = temp->next;
        }
        std::cout << std::endl;
    }

    // Function to reverse the doubly linked list
    void reverse() {
        Node* current = head;
        Node* temp = nullptr;

        while (current) {
            temp = current->prev;
            current->prev = current->next;
            current->next = temp;
            current = current->prev;
        }

        if (temp) {
            head = temp->prev;
        }
    }
};
```

```
int main() {
    // Create a doubly linked list
    DoublyLinkedList myList;
    myList.insert(1);
    myList.insert(2);
    myList.insert(3);
    myList.insert(4);

    // Print the original list
    std::cout << "Original list: ";
    myList.print();

    // Reverse the list
    myList.reverse();

    // Print the reversed list
    std::cout << "Reversed list: ";
    myList.print();

    return 0;
}
```

Running of the Program:

```
Microsoft Visual Studio Debu

Original list: 1 2 3 4
Reversed list: 4 3 2 1

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1Task1\N
To automatically close the console when debugging s
le when debugging stops.
Press any key to close this window . . .
```

Problem 3

Main.cpp

```cpp
//Naafiul Hossain
//SBU ID: 115105623

#include <iostream>

class Node {
public:
    int data;
    Node* prev;
    Node* next;

    Node(int value, Node* p = nullptr, Node* n = nullptr)
        : data(value), prev(p), next(n) {}
};

Node* removeDuplicates(Node* head) {
    Node* current = head;

    while (current != nullptr && current->next != nullptr) {
        if (current->data == current->next->data) {
            Node* nextUnique = current->next->next;
            delete current->next;
            current->next = nextUnique;
            if (nextUnique != nullptr) {
                nextUnique->prev = current;
            }
        }
        else {
            current = current->next;
        }
    }

    return head;
}
```

```cpp
void printList(Node* head) {
    Node* current = head;
    while (current != nullptr) {
        std::cout << current->data << " <=> ";
        current = current->next;
    }
    std::cout << "nullptr" << std::endl;
}

int main() {
    // Example usage:
    // Create a sorted doubly linked list: 1 <=> 2 <=> 2 <=> 3 <=> 4 <=> 4 <=> 4 <=> 5
    Node* head = new Node(1);
    head->next = new Node(2, head);
    head->next->next = new Node(2, head->next);
    head->next->next->next = new Node(3, head->next->next);
    head->next->next->next->next = new Node(4, head->next->next->next);
    head->next->next->next->next->next = new Node(4, head->next->next->next->next);
    head->next->next->next->next->next->next = new Node(4, head->next->next->next->next->next);
    head->next->next->next->next->next->next->next = new Node(5, head->next->next->next->next->next-

    std::cout << "Original Doubly Linked List:" << std::endl;
    printList(head);

    head = removeDuplicates(head);

    std::cout << "\nDoubly Linked List after Removing Duplicates:" << std::endl;
    printList(head);

    // Clean up memory
    Node* current = head;
```

```cpp
    head = removeDuplicates(head);

    std::cout << "\nDoubly Linked List after Removing Duplicates:" << std::endl;
    printList(head);

    // Clean up memory
    Node* current = head;
    while (current != nullptr) {
        Node* next = current->next;
        delete current;
        current = next;
    }

    return 0;
}
```

Screenshot of the running program:

```
Microsoft Visual Studio Debu

Original Doubly Linked List:
1 <=> 2 <=> 2 <=> 3 <=> 4 <=> 4 <=> 4 <=> 5 <=> nullptr

Doubly Linked List after Removing Duplicates:
1 <=> 2 <=> 3 <=> 4 <=> 5 <=> nullptr

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1Task1\NHLab10P3\x64\
To automatically close the console when debugging stops, enable
le when debugging stops.
Press any key to close this window . . .
```

Problem 4

Main.cpp

```cpp
//Naafiul Hossain
//SBU ID: 115107623
#include <iostream>
#include <vector>

using namespace std;

// Function to heapify a subtree rooted at given index
void maxHeapify(vector<int>& vec, int n, int root) {
    int largest = root; // Initialize largest as root
    int left = 2 * root + 1; // left child
    int right = 2 * root + 2; // right child

    // If left child is larger than root
    if (left < n && vec[left] > vec[largest])
        largest = left;

    // If right child is larger than largest so far
    if (right < n && vec[right] > vec[largest])
        largest = right;

    // If largest is not root
    if (largest != root) {
        swap(vec[root], vec[largest]);

        // Recursively heapify the affected sub-tree
        maxHeapify(vec, n, largest);
    }
}
```

```cpp
// Main function to do heap sort
void maxHeapSort(vector<int>& vec, int n) {
    // Build max heap
    for (int i = n / 2 - 1; i >= 0; i--)
        maxHeapify(vec, n, i);

    // Extract elements from the heap one by one
    for (int i = n - 1; i > 0; i--) {
        swap(vec[0], vec[i]); // Move current root to end
        maxHeapify(vec, i, 0); // call max heapify on the reduced heap
    }
}

// Function to print the top k largest numbers in ascending order
void printTopKLargest(vector<int>& vec, int n, int k) {
    maxHeapSort(vec, n); // First, sort the entire array using max heap sort

    cout << "Top " << k << " largest numbers: ";
    for (int i = n - 1; i >= max(0, n - k); i--) {
        cout << vec[i] << " ";
    }
    cout << endl;
}
```

```
int main() {
    int length, lower_bound, upper_bound;

    // Input length, lower bound, and upper bound
    cout << "Input length, lower bound, and upper bound: ";
    cin >> length >> lower_bound >> upper_bound;

    // Generate a vector of random numbers within the specified range
    vector<int> vec(length);
    for (int i = 0; i < length; i++) {
        vec[i] = rand() % (upper_bound - lower_bound + 1) + lower_bound;
    }

    // Display the original vector
    cout << "Original Vector: ";
    for (int num : vec) {
        cout << num << " ";
    }
    cout << endl;

    int k;
    // Input K for the top K largest
    cout << "Input K for the top K largest: ";
    cin >> k;

    printTopKLargest(vec, length, k);

    return 0;
}
```

Screenshot of the Program running:

```
Microsoft Visual Studio Debu

Input length, lower bound, and upper bound: 10 1 10
Original Vector: 2 8 5 1 10 5 9 9 3 5
Input K for the top K largest: 3
Top 3 largest numbers: 10 9 9

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1Task1\NHLab1
To automatically close the console when debugging stops,
le when debugging stops.
Press any key to close this window . . .
```

Problem 5

Main.cpp

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

#include <iostream>
#include <climits>

class Node {
public:
    int data;
    Node* left;
    Node* right;

    Node(int data) {
        this->data = data;
        left = NULL;
        right = NULL;
    }
};

bool isMaxHeap(Node* root) {
    if (root == NULL) {
        return true;
    }

    // Check the max heap property at the current node
    if (root->left != NULL && root->data < root->left->data) {
        return false;
    }

    if (root->right != NULL && root->data < root->right->data) {
        return false;
    }
```

```
    }

    // Recursively check the max heap property for left and right subtrees
    if (!isMaxHeap(root->left) || !isMaxHeap(root->right)) {
        return false;
    }

    return true;
}

int main() {
    Node* root = new Node(10);
    root->left = new Node(9);
    root->right = new Node(8);
    root->left->left = new Node(7);
    root->left->right = new Node(6);
    root->right->left = new Node(5);
    root->right->right = new Node(4);
    root->left->left->left = new Node(3);
    root->left->left->right = new Node(2);
    root->left->right->left = new Node(1);

    if (isMaxHeap(root)) {
        std::cout << "The binary tree is a max heap." << std::endl;
    }
    else {
        std::cout << "The binary tree is not a max heap." << std::endl;
    }

    return 0;
}
```

Screenshot of the running program:

```
Microsoft Visual Studio Debu

The binary tree is a max heap.

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1Task1\
To automatically close the console when debugging
le when debugging stops.
Press any key to close this window . . .
```

Problem 6 Extra Credit

Main.cpp

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

#include <iostream>
#include <vector>
#include <queue>

using namespace std;

int kthLargestPQ(vector<int>& vec, int k) {
    priority_queue<int, vector<int>, greater<int>> minHeap;

    for (int num : vec) {
        minHeap.push(num);
        if (minHeap.size() > k) {
            minHeap.pop();
        }
    }

    return minHeap.top();
}

int jthSmallestPQ(vector<int>& vec, int j) {
    priority_queue<int> maxHeap;

    for (int num : vec) {
        maxHeap.push(num);
        if (maxHeap.size() > j) {
            maxHeap.pop();
        }
    }

    return maxHeap.top();
}
```

```
    return maxHeap.top();
}

int main() {
    vector<int> vector3 = { 989, 2, 3, 7, 9, 5, 7, 7 };
    vector<int> vector4 = { 96, 56, 7, 7, 6, 1, 6, 2 };

    int k;
    cout << "Original vector3: ";
    for (int num : vector3) {
        cout << num << " ";
    }
    cout << "\nInput K for Kth largest: ";
    cin >> k;

    int kthLargest = kthLargestPQ(vector3, k);
    cout << "The " << k << " largest number: " << kthLargest << "\n";

    int j;
    cout << "\nOriginal vector4: ";
    for (int num : vector4) {
        cout << num << " ";
    }
    cout << "\nInput J for Jth smallest: ";
    cin >> j;

    int jthSmallest = jthSmallestPQ(vector4, j);
    cout << "The " << j << " smallest number: " << jthSmallest << "\n";

    return 0;
}
```

Running solution:

```
Original vector3: 989 2 3 7 9 5 7 7
Input K for Kth largest: 2
The 2 largest number: 9

Original vector4: 96 56 7 7 6 1 6 2
Input J for Jth smallest: 5
The 5 smallest number: 7
```