|  |
| --- |
| **Course**: Programming Fundamental – ENSF 337  **Lab #**: Lab 8  **Instructor**: M. Moussavi  **Student Name**: Aarushi Roy Choudhury  **Lab Section**: B01  **Date submitted**: Nov, 25 2021 |

**Exercise A**

Diagram

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram

Description automatically generated

Exercise B

// OLList.cpp

// ENSF 337 Fall 2021 Lab 8 Exercise A and B

// Completed by: Aarushi Roy Choudhury

#include <iostream>

#include <stdlib.h>

using namespace std;

#include "OLList.h"

OLList::OLList()

        : headM(0) {

}

OLList::OLList(const OLList &source) {

    copy(source);

}

OLList &OLList::operator=(const OLList &rhs) {

    if (this != &rhs) {

        destroy();

        copy(rhs);

    }

    return \*this;

}

OLList::~OLList() {

    destroy();

}

void OLList::print() const {

    cout << '[';

    if (headM != 0) {

        cout << ' ' << headM->item;

        for (const Node \*p = headM->next; p != 0; p = p->next)

            cout << ", " << p->item;

    }

    cout << " ]\n";

}

void OLList::insert(const ListItem &itemA) {

    Node \*new\_node = new Node;

    new\_node->item = itemA;

    if (headM == 0 || itemA <= headM->item) {

        new\_node->next = headM;

        headM = new\_node;

// point one

    } else {

        Node \*before = headM; // will point to node in front of new node

        Node \*after = headM->next; // will be 0 or point to node after new node

        while (after != 0 && itemA > after->item) {

            before = after;

            after = after->next;

        }

        new\_node->next = after;

        before->next = new\_node;

// point two

    }

}

void OLList::remove(const ListItem &itemA) {

// if list is empty, do nothing

    if (headM == 0 || itemA < headM->item)

        return;

    Node \*doomed\_node = 0;

    if (itemA == headM->item) {

        doomed\_node = headM;

        headM = headM->next;

        delete doomed\_node;

    } else {

        Node \*before = headM;

        Node \*maybe\_doomed = headM->next;

        while (maybe\_doomed != 0 && itemA > maybe\_doomed->item) {

            before = maybe\_doomed;

            maybe\_doomed = maybe\_doomed->next;

        }

        if (maybe\_doomed != nullptr) {

            before->next = maybe\_doomed->next;

            delete maybe\_doomed;

        }

// point three

    }

// the remaining part of this function is missing. As part of exercise B

// students are supposed to complete the rest of the definition of this function.

}

void OLList::destroy() {

// this function is not properly designed. As part of the exercise B

// students are supposed to remove the following lines and

// complete the definition of this helper function.

    Node\* currentNode = headM;

    while (currentNode != nullptr) {

        Node\* nextNode = currentNode->next;

        delete currentNode;

        currentNode = nextNode;

    }

    headM = nullptr;

}

void OLList::copy(const OLList &source) {

// this function is not properly designed. As part of the exercise B

// students are supposed to remove the following lines and

// complete the definition of this helper function.

// The only effect of the next line is to tell the compiler

// not to generate an "unused argument" warning. Don't leave it

// it in your solution.

    if (source.headM != nullptr) {

        Node \*newNode = new Node;

        newNode->item = source.headM->item;

        newNode->next = nullptr;

        headM = newNode;

        Node \*sourceTempNode = source.headM->next;

        Node \*currentTempNode = headM;

        while (sourceTempNode != nullptr) {

            Node \*newTempNode = new Node;

            newTempNode->item = sourceTempNode->item;

            newTempNode->next = nullptr;

            currentTempNode->next = newTempNode;

            currentTempNode = newTempNode;

            sourceTempNode = sourceTempNode->next;

        }

    } else {

        headM = nullptr;

    }

}

Text

Description automatically generated

**Exercise C**

Please see the zip file for the code, outputs for each option are provided below. I also printed the data after insertions and deletions to show they were done successfully.

Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generated