**Lab 5**

**Name: Coral S. Schmidt Montilla ID#: 148830**

**Name: Aleysha M. Rivera Cores ID#: 108408**

1. Copy the source code developed for Lab 5 and paste it as **text** below. (*15 points*)

/\*

\* CECS 2223, Computer Programming II Laboratory

\* Fall 2023

\* Date: September 27, 2023

\* Topic: Lab 5 - Composition and Dynamic Memory

\* File name: Catalog.h

\* This file declares a class named Catalog

\* Complete the declaration as required.

\* Name: Coral S. Schmidt Montilla, ID#148830

\* Name: Aleysha M. Rivera Cores, ID#108408

\*/

#pragma once

// preprocessor directives

#include "Course.h"

class Catalog {

private:

string universityName; // the name of the university

int courseCount; // the number of courses in the catalog

Course\*\* courses; // the array which contains the courses

void sortCatalog(); // sorts the courses in the catalog in alphabetical order

public:

Catalog(); // the default constructor

Catalog(const Catalog&); // the copy constructor

~Catalog(); // the destructor

void setUniversityName(string);

string getUniversityName() const;

void addCourse(string, string, int); // adds a course to the catalog

// The addCourse method calls the sortCatalog method after a course is added.

int findCourse(const string) const; // method to find a course in the catalog

void deleteCourse(int); // removes a course from the catalog

void printCatalog() const; // prints the course list for the university

// printCatalog prints the header CODE, NAME, CREDITS using the column widths

// from the Course class

void operator=(const Catalog&); // overload of the assignment operator

};

/\*

\* CECS 2223, Computer Programming II Laboratory

\* Fall 2023

\* Date: September 27, 2023

\* Topic: Lab 5 - Composition and Dynamic Memory

\* File name: Course.h

\* This file declares a class named Course

\* Complete the declaration as required.

\* Name: Coral S. Schmidt Montilla ID#148830

\* Name: Aleysha M. Rivera Cores, ID#108408

\*/

#pragma once

// preprocessor directives

#include <string>

#include <iostream>

using namespace std;

class Course {

private:

string courseCode; // the course's code, e.g. CECS 2223

string courseName; // the course's name, e.g. Computer Programming II Laboratory

int credits; // the number of credits for the course

static size\_t codeSize; // the size of the courseCode field

void setCodeSize(string, bool); // private method to set the value for codeSize

static size\_t nameSize; // the size of the courseName field

void setNameSize(string, bool); // private method to set the value for nameSize

public:

Course(); // the default constructor

Course(string, string, int); // parameterized constructor

// the parameterized constructor must validate the value for credits

Course(const Course&); // the copy constructor

~Course(); // the destructor

void setCourseCode(string);

void setCourseName(string);

void setCourseCredits(int); // validates the parameter to be 0 or greater

string getCourseCode() const;

string getCourseName() const;

int getCourseCredits() const;

size\_t getCodeSize() const;

size\_t getNameSize() const;

void displayCourse() const; // prints the course data ready for a table

// the order is course code, course name, credits, and the size of the

// course and name fields is variable

};

/\*

\* CECS 2223, Computer Programming II Laboratory

\* Fall 2023

\* Date: September 27, 2023

\* Topic: Lab 5 - Composition and Dynamic Memory

\* File name: Catalog.cpp

\* This file defines a class named Catalog

\* Complete the declaration as required.

\* Name: Coral S. Schmidt Montilla, ID#148830

\* Name: Aleysha M. Rivera Cores, ID#108408

\*/

#include "Catalog.h"

Catalog::Catalog() {

universityName = "";

courseCount = 0;

courses = nullptr;

}

Catalog::Catalog(const Catalog& other) {

universityName = other.universityName;

courseCount = other.courseCount;

if (courseCount > 0) {

courses = new Course \* [courseCount];

for (int i = 0; i < courseCount; i++) {

courses[i] = new Course(\*(other.courses[i]));

}

}

else {

courses = nullptr;

}

}

Catalog::~Catalog() {

for (int i = 0; i < courseCount; i++) {

delete courses[i];

}

delete[] courses;

}

void Catalog::setUniversityName(string name) {

universityName = name;

}

string Catalog::getUniversityName() const {

return universityName;

}

void Catalog::addCourse(string code, string name, int credits) {

Course\* newCourse = new Course(code, name, credits);

Course\*\* temp = courses;

courses = new Course \* [courseCount + 1];

for (int i = 0; i < courseCount; i++) {

courses[i] = temp[i];

}

courses[courseCount] = newCourse;

courseCount++;

delete[] temp;

// Sort the catalog after adding the new course

sortCatalog();

}

int Catalog::findCourse(const string code) const {

for (int i = 0; i < courseCount; i++) {

if (courses[i]->getCourseCode() == code) {

return i;

}

}

return -1;

}

void Catalog::deleteCourse(int index) {

if (index >= 0 && index < courseCount) {

delete courses[index];

for (int i = index; i < courseCount - 1; i++) {

courses[i] = courses[i + 1];

}

courseCount--;

Course\*\* temp = courses;

courses = new Course \* [courseCount];

for (int i = 0; i < courseCount; i++) {

courses[i] = temp[i];

}

delete[] temp;

}

}

void Catalog::printCatalog() const {

// Calculate the maximum code and name lengths

size\_t maxCodeLength = 0;

size\_t maxNameLength = 0;

for (int i = 0; i < courseCount; i++) {

size\_t codeLength = courses[i]->getCourseCode().length();

size\_t nameLength = courses[i]->getCourseName().length();

if (codeLength > maxCodeLength) {

maxCodeLength = codeLength;

}

if (nameLength > maxNameLength) {

maxNameLength = nameLength;

}

}

// Print the header

printf("University %s\n\n%-\*s %-\*s %s\n\n", universityName.c\_str(), static\_cast<int>(maxCodeLength), "CODE", static\_cast<int>(maxNameLength), "NAME", "CREDITS");

// Print course details

for (int i = 0; i < courseCount; i++) {

printf("%-\*s %-\*s %5d\n\n", static\_cast<int>(maxCodeLength),

courses[i]->getCourseCode().c\_str(), static\_cast<int>(maxNameLength),

courses[i]->getCourseName().c\_str(),

courses[i]->getCourseCredits());

}

}

void Catalog::sortCatalog() {

if (courseCount <= 1) {

return; // No need to sort if there are 0 or 1 courses

}

for (int i = 0; i < courseCount - 1; i++) {

for (int j = 0; j < courseCount - i - 1; j++) {

// Compare course names character by character

string name1 = courses[j]->getCourseName();

string name2 = courses[j + 1]->getCourseName();

int minLength = min(name1.length(), name2.length());

int k = 0;

while (k < minLength && name1[k] == name2[k]) {

k++;

}

if (k < minLength && name1[k] > name2[k]) {

Course\* temp = courses[j];

courses[j] = courses[j + 1];

courses[j + 1] = temp;

}

}

}

}

void Catalog::operator=(const Catalog& other) {

if (this == &other) {

return; // Check for self-assignment

}

// Delete existing courses

for (int i = 0; i < courseCount; i++) {

delete courses[i];

}

delete[] courses;

// Copy data from other Catalog

universityName = other.universityName;

courseCount = other.courseCount;

// Create new courses array and copy courses

if (courseCount > 0) {

courses = new Course \* [courseCount];

for (int i = 0; i < courseCount; i++) {

courses[i] = new Course(\*(other.courses[i]));

}

}

else {

courses = nullptr;

}

}

/\*

\* CECS 2223, Computer Programming II Laboratory

\* Fall 2023

\* Date: September 27, 2023

\* Topic: Lab 5 - Composition and Dynamic Memory

\* File name: Course.cpp

\* This file declares a class named Catalog

\* Complete the declaration as required.

\* Name: Aleysha M. Rivera Cores, ID# 108408

\* Name: Coral S. Schmidt Montilla, ID# 148830

\*/

#include "Course.h"

#include <iostream>

#include <string>

#include <iomanip>

using namespace::std;

//d.constructor

Course::Course() : credits(0) {

setCourseCode("");

setCourseName("");

}

//p.constructor

Course::Course(string code, string name, int credits) {

setCourseCode(code);

setCourseName(name);

setCourseCredits(credits);

}

//copy constructor

Course::Course(const Course& other) {

courseCode = other.courseCode;

courseName = other.courseName;

credits = other.credits;

}

//destructor

Course::~Course()

{ }

//set codeSize based on courseCode

void Course::setCodeSize(string, bool forceUpdate) {

if (forceUpdate || courseCode.size() > codeSize) {

codeSize = courseCode.size();

}

}

//nameSize based on courseName

void Course::setNameSize(string name, bool forceUpdate) {

if (forceUpdate || name.size() > nameSize) {

nameSize = name.size();

}

}

//Setter for crouseCode

void Course::setCourseCode(string code) {

courseCode = code;

setCodeSize(code, false); //update codeize if needed

}

//Setter for courseName

void Course::setCourseName(string name) {

courseName = name;

setNameSize(name, false); //update nameSize if needed

}

//setter for courseCredits

void Course::setCourseCredits(int c) {

if (c >= 0) {

credits = c;

}

else {

cout << "Invalid credits value. Setting to 0." << endl;

credits = 0;

}

}

//Getter for courseCode

string Course::getCourseCode() const {

return courseCode;

}

//Getter for courseName

string Course::getCourseName() const {

return courseName;

}

//Getter for courseCredits

int Course::getCourseCredits() const {

return credits;

}

//Getter for codeSize

size\_t Course::getCodeSize() const {

return codeSize;

}

//Getter for nameSize

size\_t Course::getNameSize() const {

return nameSize;

}

//Display course info

void Course::displayCourse() const {

cout << left << setw(codeSize) << courseCode << " | "

<< setw(nameSize) << courseName << " | " << setw(10) << credits << endl;

}

/\*

\* CECS 2223, Computer Programming II Laboratory

\* Fall 2023

\* Date: September 27, 2023

\* Topic: Lab 5 - Composition and Dynamic Memory

\* File name: lab05.cpp

\* This file implements a class named Catalog

\* Complete the code as required.

\* Name: Aleysha M. Rivera Cores, ID# 108408

\* Name: Coral S. Schmidt Montilla, ID# 148830

\*

\* To test your code, add at least 4 courses to the catalog, and remove at least 2.

\* Start by selecting the option to print the catalog, then proceed to add courses,

\* and finally remove courses. You should print the course list after every add or

\* remove action.

\*/

// preprocessor directives

#include "Catalog.h"

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

// Initialize static variables

size\_t Course::codeSize = 0;

size\_t Course::nameSize = 0;

string code, name;

int credits;

int menu() {

int option = 0;

cout << "Menu:" << endl;

cout << "1. View all courses in the catalog" << endl;

cout << "2. Add a course to the catalog" << endl;

cout << "3. Remove a course from the catalog" << endl;

cout << "4. Exit the program" << endl;

cout << "Enter your choice: ";

cin >> option;

return option;

}

bool execute(Catalog& catalog, const int option) {

switch (option) {

case 1:

cout << "Course Catalog:" << endl;

catalog.printCatalog();

break;

case 2: {

cout << "Enter course code: ";

cin >> code;

cout << "Enter course name: ";

cin.ignore();

getline(cin, name);

cout << "Enter course credits: ";

cin >> credits;

catalog.addCourse(code, name, credits);

catalog.printCatalog();

} break;

case 3: {

cout << "Enter course code to remove: ";

cin >> code;

int index = catalog.findCourse(code);

if (index != -1) {

catalog.deleteCourse(index);

cout << "Course " << code << " removed." << endl;

}

else {

cout << "Course " << code << " not found in the catalog." << endl;

}

catalog.printCatalog();

} break;

case 4:

return false; // Exit the program

default:

cout << "Invalid option. Please choose a valid option from the menu." << endl;

}

return true;

}

void personalInfo() {

cout << "Program developed by Aleysha M. Rivera Cores and Coral S. Schmidt Montilla." << endl << endl;

}

int main() {

Catalog poli;

poli.setUniversityName("Polytechnic University of Puerto Rico");

personalInfo();

int option;

do {

option = menu();

} while (execute(poli, option));

system("pause"); // For Visual Studio use only!

return 0;

}

// declare a Catalog object name poli

// Assign the string Polytechnic University of Puerto Rico as the name for poli

// develop a while iteration control structure which uses the return value of the

// execute method as sentinel. The execute method receive the reference to poli

// and the return value of menu as parameters.

// This must be a single line of code!

// while ();

// The menu method presents the user with a list of options. They are:

// 1. View all courses in the catalog

// 2. Add a course to the catalog

// 3. Remove a course from the catalog

// 4. Exit the program

// Declare the local integer variable option and initialize if to 0

// This method DOES NOT validate the value received by the user,

// it only returns such value.

// The execute method implements a switch selection control structure to

// implement the menu option. It's parameters are a reference to a Catalog

// object and an integer representing the option selected by the user.

// The method returns false only when the option value is 4, true otherwise.

// The default case catches invalid option values and requires an error

// message to be printed.

// The personalInfo method prints the phrase

// Program developed by [name of student 1] and [name of student 2].

// make sure to add an empty line after the phrase is printed.

1. Paste the screenshots of the program’s execution below. (*5 points*)

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

1. Comment on any warnings or errors revealed by Visual Studio. If any error messages were present, list the error and describe how you corrected it. If no errors or warnings were revealed, comment on the most important aspect of developing the solution. (*5 points*)

A screenshot of a computer

Description automatically generated

We fixed it by eliminating:

size\_t Course::codeSize = 0;

size\_t Course::nameSize = 0;

From Course.cpp