|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE101L: Object Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 02/06/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 12B. Problem Based Learning through Objects & Classes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| Muhammad Abubakar Saif | BSCE21017 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to observe the basic knowledge of programming classes in C++.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

**Open-ended problem** is a problem that has several or many correct answers, and several ways to the correct answer(s). The Open-Ended Approach provides students with "experience in finding something new in the process"(Shimada 1997). It is basically facilitating the development of creative problem solving skills.

Diagram

Description automatically generated

Figure 1: \*What is Open Ended Problem Solving??

**Lab Task**

**Task A: [Marks: 20]**

Create a class RationalNumber (fractions) with these capabilities:

a) Create a constructor that prevents a 0 denominator in a fraction, reduces or simplifies fractions that are not in reduced form and avoids negative denominators.

b) Overload the addition, subtraction, multiplication, and division operators for this class.

c) Overload the relational and equality operators for this class.

|  |
| --- |
| **(RationalNumber.h):**  #ifndef INC\_2022\_SPRING\_CE\_OOP\_WEEK12\_LABTASK\_B\_BSCE21017\_RATIONALNUMBER\_H #define INC\_2022\_SPRING\_CE\_OOP\_WEEK12\_LABTASK\_B\_BSCE21017\_RATIONALNUMBER\_H #include **<iostream>  using namespace** std;  **class** RationalNumber { **private**:  **int** numerator, denominator; **public**:  RationalNumber(**int** dummy);  RationalNumber();  **void** simplest();  RationalNumber **operator** + (RationalNumber t1) {  RationalNumber res;  res.numerator = numerator \* t1.denominator + t1.numerator \* denominator;  res.denominator = denominator \* t1.denominator;  res.simplest();  **return** res;  }  RationalNumber **operator** - (RationalNumber t1) {  RationalNumber res;  res.numerator = numerator \* t1.denominator - t1.numerator \* denominator;  res.denominator = denominator \* t1.denominator;  res.simplest();  **return** res;  }  RationalNumber **operator** \* (RationalNumber t1) {  RationalNumber result;  result.numerator = numerator \* t1.numerator;  result.denominator = denominator \* t1.denominator;  result.simplest();  **return** result;  }  RationalNumber **operator** / (RationalNumber t1) {  RationalNumber answer;  answer.numerator = numerator \* t1.denominator;  answer.denominator = denominator \* t1.numerator;  answer.simplest();  **return** answer;  }  **friend** ostream& **operator** << (ostream& os, RationalNumber &r1){  **if** (r1.numerator == 0){  cout<<r1.numerator;  } **else** cout<<r1.numerator<<**"/"**<<r1.denominator;  **return** os;  }  **bool operator** == (RationalNumber second) {  **if** (numerator == second.numerator){  **if** (denominator == second.denominator){  **return true**;  }  }  **return false**;  }  **bool operator** < (RationalNumber s1) {  **double** num1 = (**double**) numerator / (**double**) denominator;  **double** num2 = (**double**) s1.numerator / (**double**) s1.denominator;  **if** (num1 < num2){  **return true**;  }**else return false**;  }  **bool operator** > (RationalNumber s1) {  **double** num1 = (**double**) numerator / (**double**) denominator;  **double** num2 = (**double**) s1.numerator / (**double**) s1.denominator;  **if** (num1 > num2){  **return true**;  }**else return false**;  } };   #endif *//INC\_2022\_SPRING\_CE\_OOP\_WEEK12\_LABTASK\_B\_BSCE21017\_RATIONALNUMBER\_H*  **(RationalNumber.cpp):**  #include **"RationalNumber.h"** #include **<iostream>  using namespace** std;  RationalNumber::RationalNumber(**int** dummy) {  cout << **"Enter Numerator: "**;  cin >> numerator;  **do** {  cout << **"Enter Denominator: "**;  cin >> denominator;  } **while** (denominator <= 0 && cout << **"Value Entered is Zero or Negative, Enter Again \n"**);  simplest(); }  RationalNumber::RationalNumber() {  numerator = 0;  denominator = 0; }  **void** RationalNumber::simplest() {  **int** rem = 0, rem1 = 0, q1 = 0, q2 = 0, limit = 0, check = 0, act1 = 0, act2 = 0;  **if** (numerator < denominator) {  limit = numerator;  } **else** limit = denominator;  **for** (**int** i = 2; i < limit + 1; ++i) {  rem = numerator % i;  rem1 = denominator % i;  q1 = numerator / i;  q2 = denominator / i;  **if** (rem == 0 && rem1 == 0) {  act1 = q1;  act2 = q2;  check++;  }  }  **if** (check != 0) {  numerator = act1;  denominator = act2;  } }  **(MAIN FUNCTION):**  **int** main {  **int** choice;  string rep;  RationalNumber r1(1);  RationalNumber r2(1);  RationalNumber r3;  again:  cout << **"Menu \n"**;  cout << **"1. Addition of Rational Numbers \n"**;  cout << **"2. Subtraction of Rational Numbers \n"**;  cout << **"3. Multiplication of Rational Numbers \n"**;  cout << **"4. Division of Rational Numbers \n"**;  cout << **"5. Check Equality of Rational Number \n"**;  cout << **"6. Compare Rational Numbers \n"**;  cout << **"Enter your Choice: "**;  cin >> choice;  **switch** (choice) {  **case** 1:  r3 = r1 + r2;  cout << **"Sum of Given Two Rational Number: "**;  cout << r1 << **" + "** << r2 << **" = "**;  cout << r3 << endl;  **break**;  **case** 2:  r3 = r1 - r2;  cout << **"Difference of Given Two Rational Number: "**;  cout << r1 << **" - "** << r2 << **" = "**;  cout << r3 << endl;  **break**;  **case** 3:  r3 = r1 \* r2;  cout << **"Product of Given Two Rational Number: "**;  cout << r1 << **" x "** << r2 << **" = "**;  cout << r3 << endl;  **break**;  **case** 4:  cout  << **"Press 1 to use first Rational Number as Dividend and Second Rational Number as Divisor (First Number/Second Number) \n"**;  cout  << **"Press 2 to use Second Rational Number as Dividend and First Rational Number as Divisor (Second Number/First Number) \n"**;  cout << **"Enter Your Choice: "**;  cin >> choice;  **switch** (choice) {  **case** 1:  r3 = r1 / r2;  cout << **"Quotient of Given Two Rational Number: "**;  cout << r1 << **" / "** << r2 << **" = "**;  cout << r3 << endl;  **break**;  **case** 2:  r3 = r2 / r1;  cout << **"Quotient of Given Two Rational Number: "**;  cout << r2 << **" / "** << r1 << **" = "**;  cout << r3 << endl;  **break**;  }  **break**;  **case** 5:  **if** (r1 == r2) {  cout << **"Both Rational Numbers are Equal \n"**;  } **else** cout << **"Not Equal \n"**;  **break**;  **case** 6:  cout << **"1. First Rational Number < Second Rational Number \n"**;  cout << **"2. Second Rational Number < First Rational Number \n"**;  cout << **"3. First Rational Number > Second Rational Number \n"**;  cout << **"4. Second Rational Number > First Rational Number \n"**;  cout << **"Enter Your Choice: "**;  cin >> choice;  **switch** (choice) {  **case** 1:  **if** (r1 < r2) {  cout << **"First Rational Number"** << r1 << **" is smaller than Second Rational Number"** << r2  << **" \n"**;  } **else** cout << **"First Rational Number"** << r1 << **" is not smaller than Second Rational Number"** << r2  << **" \n"**;  **break**;  **case** 2:  **if** (r2 < r1) {  cout << **"Second Rational Number "** << r2 << **" is smaller than First Rational Number"** << r1  << **" \n"**;  } **else** cout << **"Second Rational Number "** << r2 << **" is not smaller than First Rational Number"** << r1  << **" \n"**;  **break**;  **case** 3:  **if** (r1 > r2) {  cout << **"First Rational Number "** << r1 << **" is larger than Second Rational Number"** << r2  << **" \n"**;  } **else** cout << **"First Rational Number "** << r1 << **" is not larger than Second Rational Number"** << r2  << **" \n"**;  **break**;  **case** 4:  **if** (r2 > r1) {  cout << **"Second Rational Number "** << r2 << **" is larger than First Rational Number"** << r1  << **" \n"**;  } **else** cout << **"Second Rational Number "** << r2 << **" is not larger than First Rational Number"** << r1  << **" \n"**;  **break**;  }  **break**;  }  cout << **"Do you want to use the Rational Number Calculator again? (Y/N): "**;  cin >> rep; *//takes input from user* **if** (rep == **"N" or** rep == **"n" or** rep == **"no" or** rep == **"NO" or** rep == **"No" or** rep == **"nO"**) {  **return 0**; } **else goto** again; *//restart the program flow* } |

**Output:**

**Input & Addition:**

**Text

Description automatically generated**

**Subtraction:**

**Text, letter

Description automatically generated**

**Multiplication:**

**Text, letter

Description automatically generated**

**Division:**

**Graphical user interface, text, application, email

Description automatically generated**

**Checking Equality:**

**Text

Description automatically generated**

**Comparing Numbers:**

**Text, letter

Description automatically generated**

****

****

****

**Task B: [Marks: 20]**

(Polynomial Class) Develop class Polynomial. The internal representation of a Polynomial is an array of terms. Each term contains a coefficient and an exponent, e.g., the term 2x4 has the coefficient 2 and the exponent 4. Develop a complete class containing proper constructor and destructor functions as well as set and get functions. The class should also provide the following overloaded operator capabilities:

a) Overload the addition operator (+) to add two Polynomials.

b) Overload the subtraction operator (-) to subtract two Polynomials.

c) Overload the assignment operator to assign one Polynomial to another.

d) Overload the multiplication operator (\*) to multiply two Polynomials.

e) Overload the addition assignment operator (+=), subtraction assignment operator (-=), and multiplication assignment operator (\*=).

|  |
| --- |
| **(Polynomial.h):**  #ifndef INC\_2022\_SPRING\_CE\_OOP\_WEEK12\_LABTASK\_B\_BSCE21017\_POLYNOMIAL\_H #define INC\_2022\_SPRING\_CE\_OOP\_WEEK12\_LABTASK\_B\_BSCE21017\_POLYNOMIAL\_H  #include **<iostream>** #include **<string>  using namespace** std;  **class** Polynomial { **private**:  **int** \*coefficient;  **int** degree; **public**:  Polynomial(){}  Polynomial(**int** power);  **void** setCoefficients(**int** \*arr, **int** size = 0);  **int** \*getEquation();  **friend** istream &**operator**>>(istream &in, Polynomial &test);  **friend** ostream &**operator**<<(ostream &out, Polynomial &test) {  **int** pseudo = test.degree - 1;  **for** (**int** i = 0; i < test.degree; ++i) {  **if** (i < test.degree - 2 && test.coefficient[i] != 0) {  **if** (test.coefficient[i] < 0) {  cout << **"("** << test.coefficient[i] << **"x^"** << pseudo << **")"**;  } **else** cout << test.coefficient[i] << **"x^"** << pseudo;  **if** (test.coefficient[i + 1] != 0)  cout << **" + "**;  } **else** {  **if** (i == test.degree - 1 && test.coefficient[i] != 0) {  **if** (test.coefficient[i] < 0) {  cout << **"("** << test.coefficient[i] << **")"**;  } **else** cout << test.coefficient[i];  }  **if** (i == test.degree - 2 && test.coefficient[i] != 0) {  **if** (test.coefficient[i] < 0) {  cout << **"("** << test.coefficient[i] << **"x"** << **")"**;  } **else** cout << test.coefficient[i] << **"x"**;  **if** (test.coefficient[i + 1] != 0)  cout<<**" + "**;  }  }  pseudo--;  }  **return** out;  }  Polynomial **operator** + (Polynomial num1);  Polynomial **operator** - (Polynomial num1);  Polynomial **operator** \* (Polynomial num1);   **void operator** = (Polynomial num1) {  degree = num1.degree;  setCoefficients(num1.coefficient);  **return**;  }  ~Polynomial(); };  **bool** isNumber(**const** string &str);  #endif *//INC\_2022\_SPRING\_CE\_OOP\_WEEK12\_LABTASK\_B\_BSCE21017\_POLYNOMIAL\_H*  **(Polynomial.cpp):**  #include **"Polynomial.h"** #include **<iostream>** #include **<string>  using namespace** std;  Polynomial::~Polynomial() {  **delete**[] coefficient; }  Polynomial::Polynomial(**int** power) {  degree = power + 1;  coefficient = **new int**[degree](); }  **void** Polynomial::setCoefficients(**int** \*arr, **int** size) {  copy:  **if** (coefficient) {  **for** (**int** i = 0; i < degree; ++i) {  coefficient[i] = arr[i];  }  } **else** {  cout<<**"Entered \n"**;  coefficient = **new int** [degree]();  **goto** copy;  } }  **int** \* Polynomial::getEquation() {  **return** coefficient; }  **bool** isNumber(**const** string& str) {  **for** (**char const** &c : str) {  **if** (isdigit(c) == 0) **return false**;  }  **return true**; }  istream &**operator**>>(istream &in, Polynomial &p1) {  **int** pDeg, \*arr, test1;  string test;  arr = **new int**[p1.degree]();  pDeg = p1.degree - 1; *// Polynomial p1(p1.degree);* cout << **"Note: If you want to skip all the next values at any instance, Enter 'E' at that point \n"**;  **do** {  cout << **"Enter Coefficient of Exponent "** << pDeg << **" (x^"** << pDeg << **") :"**;  cin >> test;  **if** (isNumber(test)) {  test1 = stoi(test);  arr[(p1.degree-1) - pDeg] = test1; *// cout<<(p1.degree-1)-pDeg<<endl;* pDeg--;  } *// cout<<test<<endl;* } **while** (pDeg >= 0 && test != **"E"**);  p1.setCoefficients(arr);  **return** in; }  Polynomial Polynomial::**operator**+(Polynomial num1) { *// Polynomial result;* **if** (degree == num1.degree){ *// result.degree = degree; // cout<<result.degree; // result.coefficient = new int [result.degree]; // for (int i = 0; i < degree; ++i) { // result.coefficient[i] = coefficient[i] + num1.coefficient[i]; // }* } *// return result;* }  **(MAIN FUNCTION):**  */\*#################POLYNOMIAL DRIVER CODE########################\*/* **int** main {  **int** deg;  cout << **"Write the Degree (Highest Power) of First Polynomial equation: "**;  cin >> deg;  Polynomial p1(deg);  cin >> p1;  cout << **"Write the Degree (Highest Power) of Second Polynomial equation: "**;  cin >> deg;  Polynomial p2(deg);  cin >> p2; *// Polynomial p3;* cout << **"First Equation: "** << p1 << endl << **"\n Second Equation: "** << p2 << endl;  **return 0**;  } |

Output:

Taking Input & Displaying it:

Text

Description automatically generated with medium confidence

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & GitHub Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_