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

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 18/05/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 10A. Problem Based Learning through Open Ended Questions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| NIMRA MAQBOOL | BSCE21012 |  |  |  |

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]**

Write a program to print the sum, difference and product of two complex numbers by creating a class named 'Complex' with separate methods for each operation whose real and imaginary parts are entered by user

|  |
| --- |
| **FUNCTION.H:**  #include <iostream> #include <string> using namespace std;  class Complex { private:  double real;  double imaginary; //declaring public:    Complex() {  int r = 0;  int imag = 0; //equaling to zero  real = r;  imaginary = imag;  cout << "DEFAULT CONSTRUCTOR IS CALLED." << endl; //displaying  }   Complex(double r, double imag) { //making parametrized constructor  cout << "ENTER REAL PART = "; //taking input  cin >> r;  cout << "ENTER IMAGINARY PART = ";  cin >> imag;  real = r;  imaginary = imag; //copying  }   void multiplyComplexNumber(Complex C, Complex C1); //declaring    void addComplexNumber(Complex C, Complex C1);   void subComplexNumber(Complex C, Complex C1); };  **FUNCTION.CPP:**  #include "Complex.h" #include <iostream>  using namespace std;  void Complex::multiplyComplexNumber(Complex C, Complex C1) {  real = C.real \* C1.real; //multiplying both by the objects passed in the parameter  imaginary = C.imaginary \* C1.imaginary;  if (imaginary < 0) { //condition that if it is smaller  cout << "MULTIPLICATION = " << real << imaginary << "i" << endl;  } else {   cout << "MULTIPLICATION = " << real << " + " << imaginary << "i" << endl;  } }  void Complex::addComplexNumber(Complex C, Complex C1) {  real = C.real + C1.real; //copying  imaginary = C.imaginary + C1.imaginary;  cout << "THE SUM = " << real << " + " << imaginary << "i" << endl; //displaying }  void Complex::subComplexNumber(Complex C, Complex C1) {  real = C.real - C1.real;  imaginary = C.imaginary - C1.imaginary; //copying  if (imaginary < 0) {  cout << "SUBTRACTION = " << real << imaginary << "i" << endl;  } else { //displaying   cout << "SUBTRACTION = " << real << " + " << imaginary << "i" << endl;  } }  **MAIN.CPP:**  #include <iostream> #include "Complex.h" #include <string>  using namespace std;  int main() {  double r;  double imag; //declaring  Complex D; //makimg object  int opt;  do {  cout << "TASK 1 OR TASK 2?" << endl;  cout << "1.TASK 1" << endl; //taking choice  cout << "2.TASK 2" << endl;  cout << "3.exit.." << endl;  cin >> opt;  if (opt == 1) {  do {  cout << "WHICH TASK ?" << endl;  cout << "1.SUBTRACTION." << endl;  cout << "2.ADDITION." << endl; //taking choice  cout << "3.MULTIPLICATION." << endl;  cout << "4.EXIT.." << endl;  cin >> opt;  if (opt == 1) {  Complex c1; //making object  Complex C(r, imag); //calling parametrized constructor  Complex C1(r, imag);  c1.subComplexNumber(C, C1); //calling function  }  if (opt == 2) {  Complex d; //making object  Complex C(r, imag);  Complex C1(r, imag); //calling parametrized constructor  d.addComplexNumber(C, C1); //calling function  }  if (opt == 3) {  Complex c;  Complex C(r, imag);  Complex C1(r, imag); //calling parametrized constructor  c.multiplyComplexNumber(C, C1); //calling function  }  if (opt == 4) {  cout << "YOU CHOOSE TO EXIT.." << endl; //exiting  exit(4);  }  } while (opt >= 1 && opt <= 4);  **OUTPUT:**  **Text  Description automatically generated** |

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

Write a program in which you have to create a class that has variables to store Car data like; CarModel, CarName, CarPrice and CarOwner. The program should include functions to assign user defined values to the above-mentioned variable and a display function to show the values. Write a main that calls these functions. Now write another runner class that declares three Car objects and displays the data of all three.

|  |
| --- |
| **FUNCTION.H:**  class Car{ private:  int CarModel;  int CarPrice; //declaring  string OwnerName;  string CarName; public:  Car(){  int cM;  int cP; //declaring  string oN;  string cN;  cout<<"ENTER CAR MODEL = ";  cin>> cM; //taking input  CarModel=cM;  cout<<"ENTER CAR PRICE = ";  cin>>cP; //taking input  CarPrice=cP; //copying  cout<<"ENTER CAR OWNER NAME = ";  cin>> cN; //taking input  OwnerName=cN;  cout<<"ENTER CAR NAME = ";  cin>> oN; //taking input  CarName=oN; //copying    }  Car(int cM,int cP,string oN,string cN){  CarModel=cM;  CarPrice=cP;  OwnerName=oN; //copying  CarName=cN;  }   void display(){  cout<<"CAR NAME = "<<CarName<<endl;  cout<<"CAR PRICE = "<<CarPrice<<endl; //displaying  cout<<"CAR MODEL = "<<CarModel<<endl;  cout<<"CAR OWNER NAME = "<<OwnerName<<endl;  } }; class CarHelper{ public:  Car c1;  Car C2; //making object of car class  Car c3;  void display(){  c1.display(); //calling the display function by the object composed  cout<<"ENTER DATA FOR OBJECT 2.."<<endl;  C2.display();  cout<<"ENTER DATA FOR OBJECT 3.."<<endl;  c3.display();  } };  #endif //INC\_2022\_SPRING\_CE\_OOP\_WEEK10\_LABTASK\_A\_BSCE21012\_COMPLEX\_H  **MAIN.CPP:**  #include <iostream> #include "Complex.h" #include <string>  using namespace std;  int main() {  double r;  double imag; //declaring  Complex D; //makimg object  int opt;  do {  cout << "TASK 1 OR TASK 2?" << endl;  cout << "1.TASK 1" << endl; //taking choice  cout << "2.TASK 2" << endl;  cout << "3.exit.." << endl;  cin >> opt;  if (opt == 1) {  do {  cout << "WHICH TASK ?" << endl;  cout << "1.SUBTRACTION." << endl;  cout << "2.ADDITION." << endl; //taking choice  cout << "3.MULTIPLICATION." << endl;  cout << "4.EXIT.." << endl;  cin >> opt;  if (opt == 1) {  Complex c1; //making object  Complex C(r, imag); //calling parametrized constructor  Complex C1(r, imag);  c1.subComplexNumber(C, C1); //calling function  }  if (opt == 2) {  Complex d; //making object  Complex C(r, imag);  Complex C1(r, imag); //calling parametrized constructor  d.addComplexNumber(C, C1); //calling function  }  if (opt == 3) {  Complex c;  Complex C(r, imag);  Complex C1(r, imag); //calling parametrized constructor  c.multiplyComplexNumber(C, C1); //calling function  }  if (opt == 4) {  cout << "YOU CHOOSE TO EXIT.." << endl; //exiting  exit(4);  }  } while (opt >= 1 && opt <= 4);  }  if (opt == 2) {  CarHelper H; //making an object  H.display(); //calling function   }  if (opt == 3) {  cout << "YOU CHOOSE TO EXIT.." << endl; //exiting  exit(4);  }  } while (opt >= 1 && opt <= 3); //condition for do while  return 0; }  **OUTPUT:**  **Text  Description automatically generated** |

#### **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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_