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

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 08/06/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 13A. 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: 40]**

A company has three types of employees:

1. Managers: receive a fixed monthly salary

2. Commission Workers: receive $1000 plus 5% of their gross monthly sales

3. Piece Workers: receive a fixed amount of money per item for each of the items they

produce; each pieceworker in this company works on only one type of item. Write a program to compute the monthly pay for each employee.

P.S: You do not know the number of employees in advance.

Each type of employee has its own pay code:

 Managers have code 1

 Commission Workers have code 2

 Pieceworkers have code 3

The user will enter the type of employee, then the program will ask about the needed information to calculate the monthly salary of this employee.

For managers, it will ask for the monthly salary of this manager.

For commission workers it will ask for the gross monthly sales.

For piece workers it will ask for the number of items produced and the price per item. When this information is entered, the program will display the salary. When the user finishes the input, the program should display the total number of employees and the total amount of money paid to these employ

|  |
| --- |
| **Function.h:**  // // Created by Lenovo on 6/9/2022. //  #ifndef INC\_2022\_SPRING\_CE\_OOP\_WEEK13\_LABTASK\_A\_BSCE21012\_FUNCTIONS\_H #define INC\_2022\_SPRING\_CE\_OOP\_WEEK13\_LABTASK\_A\_BSCE21012\_FUNCTIONS\_H  #include <iostream>  using namespace std;  class calculateSalary { public:  int salaryOfManagerJunior;  int salaryOfManagerSenior;  int salaryOfManagerMostSenior; //declaring  int payCode;  float grossMonthlySale[900];  int numberOfItemProduced[900];  int pricePerItem[900];  int numberOfCommissionedOfficer;  int numberOfPieceworkers;  int fixedMoneyToCommissionedOfficer;  int salaryOfCommissionedOfficer[900];  int salaryOfPieceWorker[900];  int totalSumOfSalary;  int totalSumOfSalary1;  int employeeNumber;   calculateSalary() {  payCode = 0;  numberOfCommissionedOfficer = 0;  numberOfPieceworkers = 0;  salaryOfManagerJunior = 0;  salaryOfManagerSenior = 0; //putting them equal to zero  salaryOfManagerMostSenior = 0;  fixedMoneyToCommissionedOfficer = 1000;  totalSumOfSalary = 0;  totalSumOfSalary1 = 0;   }   void managerSalary() {  int opt;   do {  cout << "THERE ARE 3 KINDS OF MANAGERS IN THIS COMPANY." << endl;  cout << "1.JUNIOR." << endl; //taking choices  cout << "2.SENIOR." << endl;  cout << "3.MOST SENIOR." << endl;  cout << "5.DISPLAY." << endl;  cout << "4.EXIT" << endl;  cin >> opt;  if (opt == 1) {  cout << "ENTER SALARY OF JUNIOR MANAGER = ";  cin >> salaryOfManagerJunior; //taking input  }  if (opt == 2) {  cout << "ENTER SALARY OF SENIOR MANAGER = ";  cin >> salaryOfManagerSenior;  }  if (opt == 3) {  cout << "ENTER SALARY OF MOST SENIOR MANAGER = ";  cin >> salaryOfManagerMostSenior;  }  if (opt == 5) {  cout << "YOU CHOOSE TO EXIT.." << endl;  }  if (opt == 4) {  int opt1;  do {  cout << "WHICH MANAGER?" << endl;  cout << "1.JUNIOR." << endl;  cout << "2.SENIOR." << endl;  cout << "3.MOST SENIOR." << endl; //taking choice  cout << "4.EXIT" << endl;  cin >> opt1;  if (opt1 == 1) {  cout << "SALARY OF JUNIOR MANAGER = " << salaryOfManagerJunior << endl;  }  if (opt1 == 2) {  cout << "SALARY OF SENIOR MANAGER = " << salaryOfManagerSenior << endl;  } //displaying salaries  if (opt1 == 3) {  cout << "SALARY OF MOST SENIOR MANAGER = " << salaryOfManagerMostSenior << endl;  }  if (opt1 == 4) {  cout << "YOU CHOOSE TO EXIT.." << endl;  exit(3);  }  } while (opt >= 1 && opt <= 4);   }  } while (opt >= 1 && opt <= 4);    }    void salaryOfCommissionedOfficers() {  cout << "ENTER THE NUMBER OF COMMISSIONED OFFICERS = ";  cin >> numberOfCommissionedOfficer;  cout << "ENTER THE GROSS SALE "; //taking number of officers to put the condition on loop  for (int i = 0; i < numberOfCommissionedOfficer; i++) {  cin >> grossMonthlySale[i]; //taking gross income  salaryOfCommissionedOfficer[i] = fixedMoneyToCommissionedOfficer + 0.05 \* grossMonthlySale[i]; //multiplying and then adding  totalSumOfSalary1 = totalSumOfSalary1 + salaryOfCommissionedOfficer[i]; //adding to find total  cout << "salary = " << totalSumOfSalary1 << endl; //displaying  }  }   void salaryOfPieceWorkers() {  cout << "ENTER THE NUMBER OF PIECE WORKERS = ";  cin >> numberOfPieceworkers; //taking number of officers to put the condition on loop  cout << "ENTER THE FOLLOWING INFO :";  for (int i = 0; i < numberOfPieceworkers; i++) {  cout << "ENTER NUMBER OF ITEMS PRODUCED = ";  cin >> numberOfItemProduced[i]; //taking input  cout << "ENTER PRICE PER ITEM = ";  cin >> pricePerItem[i]; //taking input  salaryOfPieceWorker[i] = numberOfItemProduced[i] \* pricePerItem[i]; //multiplying them  totalSumOfSalary = totalSumOfSalary + salaryOfPieceWorker[i]; //adding  cout << "SALARY = " << totalSumOfSalary; //displaying  }  }   void totalEmployee() {  employeeNumber = 3 + numberOfCommissionedOfficer + numberOfPieceworkers; //adding  cout << "TOTAL NUMBER OF EMPLOYEES = " << employeeNumber << endl; //displaying  cout << "TOTAL SUM OF COMMISSIONED OFFICER SALARY = " << totalSumOfSalary1 << endl;  cout << "TOTAL SUM OF PIECE WORKER SALARY = " << totalSumOfSalary << endl;  } };  **Main.cpp:**  #include <iostream> #include "Functions.h"  using namespace std;  int main() {  int opt;  calculateSalary S;   do{  cout<<"1.MANAGER"<<endl;  cout<<"2.COMMISSIONED OFFICER."<<endl; //asking for the choice  cout<<"3.PIECE WORKER ."<<endl;  cout<<"4.DISPLAY WHOLE SALARY AND EMPLOYEES."<<endl;  cout<<"5.EXIT."<<endl;  cin>>opt;  if(opt==1){  S.managerSalary();  }  if(opt==2){  S.salaryOfCommissionedOfficers();  }  if(opt==3){  S.salaryOfPieceWorkers(); //calling  }  if(opt==4){  S.totalEmployee();  }  if(opt==5){  cout<<"YOU CHOOSE TO EXIT.."<<endl;  exit(5);  }  }while(opt>=1 && opt<=4);   return 0; } |

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