| **Software Engineering Department - ITU** |
| --- |
| **SE101T: Programming Fundamentals Lab** |

| **Course Instructor: Usama Bin Shakeel** | **Dated: 30/10/2023** |
| --- | --- |
| **Teaching Assistant: Abeera Ashraf** | **Semester: Fall 2023** |
| **Teaching Assistant: Aasma Waheed** | **Batch: BSSE2023** |

# **Lab 10. DMA & Structures in C++**

***Q1. Write a program in a function named sum to add two distances in inch-feet using structure. The values of the distances are to be taken from the user and passed to the function to get the sum.***

| #include <iostream>  // Structure definition for distance  struct Distance {  int feet;  int inches;  };  // Function to add two distances  Distance sum(const Distance& distance1, const Distance& distance2) {  Distance result;  // Add the inches  result.inches = distance1.inches + distance2.inches;  // Add the feet and any additional inches  result.feet = distance1.feet + distance2.feet + (result.inches / 12);  result.inches %= 12; // Ensure inches are less than 12  return result;  }  int main() {  // Get the values for the first distance from the user  Distance distance1;  std::cout << "Enter the first distance:" << std::endl;  std::cout << "Feet: ";  std::cin >> distance1.feet;  std::cout << "Inches: ";  std::cin >> distance1.inches;  // Get the values for the second distance from the user  Distance distance2;  std::cout << "Enter the second distance:" << std::endl;  std::cout << "Feet: ";  std::cin >> distance2.feet;  std::cout << "Inches: ";  std::cin >> distance2.inches;  // Call the sum function to get the sum of distances  Distance result = sum(distance1, distance2);  // Display the result  std::cout << "Sum of distances: " << result.feet << " feet, " << result.inches << " inches." << std::endl;  return 0;  } |
| --- |

***Q2. Write a program in a function named distanceArray to create a dynamic array of Distance struct created in Q1 using the ‘new’ operator, size of array should be taken from the user. Prompt the user to enter each distance value (feet and inches) and print the distance in descending order.***

| #include <iostream>  // Structure definition for distance  struct Distance {  int feet;  int inches;  };  // Function to create a dynamic array of Distance structs and print in descending order  void distanceArray() {  int size;  // Get the size of the array from the user  std::cout << "Enter the size of the array: ";  std::cin >> size;  // Create a dynamic array of Distance structs using 'new'  Distance\* distancesArray = new Distance[size];  // Prompt the user to enter each distance value  for (int i = 0; i < size; ++i) {  std::cout << "Enter distance " << i + 1 << " (feet inches): ";  std::cin >> distancesArray[i].feet >> distancesArray[i].inches;  }  // Sort the array in descending order (based on total inches)  for (int i = 0; i < size - 1; ++i) {  for (int j = i + 1; j < size; ++j) {  int totalInchesI = distancesArray[i].feet \* 12 + distancesArray[i].inches;  int totalInchesJ = distancesArray[j].feet \* 12 + distancesArray[j].inches;  if (totalInchesI < totalInchesJ) {  // Swap the elements  std::swap(distancesArray[i], distancesArray[j]);  }  }  }  // Print the distances in descending order  std::cout << "Distances in descending order:" << std::endl;  for (int i = 0; i < size; ++i) {  std::cout << "Distance " << i + 1 << ": " << distancesArray[i].feet << " feet, "  << distancesArray[i].inches << " inches" << std::endl;  }  // Deallocate memory for the dynamic array  delete[] distancesArray;  }  int main() {  // Call the function to create a dynamic array of Distance structs and print in descending order  distanceArray();  return 0;  } |
| --- |

***Q3. Write a program in a function named employees, it will use an object of employee to manage payroll. It should include employee details (name, ID, salary) and allow for separate functions such as salary input, salary display and displaying employee information.***

| #include <iostream>  #include <string>  // Class definition for Employee  class Employee {  public:  // Member functions to set and get employee details  void setDetails(const std::string& empName, int empID);  void setSalary(double empSalary);  void displaySalary() const;  void displayInfo() const;  private:  // Member variables for employee details  std::string name;  int employeeID;  double salary;  };  // Function to input salary for an employee  void inputSalary(Employee& emp) {  double newSalary;  std::cout << "Enter the salary for employee " << emp.displayInfo() << ": ";  std::cin >> newSalary;  emp.setSalary(newSalary);  }  // Function to display salary for an employee  void displaySalary(const Employee& emp) {  std::cout << "Salary for employee " << emp.displayInfo() << ": ";  emp.displaySalary();  std::cout << std::endl;  }  // Function to display employee information  void displayEmployeeInfo(const Employee& emp) {  std::cout << "Details for employee " << emp.displayInfo() << ":" << std::endl;  emp.displaySalary();  std::cout << std::endl;  }  // Function to manage payroll using Employee objects  void employees() {  // Create an Employee object  Employee emp;  // Set employee details  emp.setDetails("John Doe", 101);  // Input and display salary for the employee  inputSalary(emp);  displaySalary(emp);  // Display employee information  displayEmployeeInfo(emp);  }  // Member function definitions for the Employee class  void Employee::setDetails(const std::string& empName, int empID) {  name = empName;  employeeID = empID;  }  void Employee::setSalary(double empSalary) {  salary = empSalary;  }  void Employee::displaySalary() const {  std::cout << "$" << salary;  }  void Employee::displayInfo() const {  std::cout << name << " (ID: " << employeeID << ")";  }  int main() {  // Call the employees function to manage payroll using Employee objects  employees();  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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_