Week 5: String Class, Array of Objects, Reference Variable, Use of Project feature in CodeBlocks IDE.
Term Project Proposal Discussion

Learning Materials: Chapter 7

Demonstration:

1. Create a Project in codeblocks IDE to Implement.
2. Use of reference variables in return type.
3. Array of objects
4. Class String

Task 1:

Create a **Student** class. An object of this class has Student Name, Department (String), Program (String), Section (String), Admission year (int), Address (type: String multiline input), and GPA (initial: 0).

Provide getter setter function for each member variable.

Provide generateID function with necessary checking (Follow the IUT ID format).

Provide a public member function **calculateGPA** that calculates the GPA for a given set of courses. Each course will contain marks for 4 quizzes, mid terms, finals and attendance. The total marks will be 100. You may create extra member variables.

Write a driver program to test the class Student.

Write a function (non-member) void **EditInformationByKeyboard**() that takes a Student object as parameter. This function will take input from the keyboard and set the member variable. Now call this function for the first 2 objects.

Create 100 **Student** objects.

Write a function (non-member) void **generateInformaiotnRandom**() takes a Student object as parameter. This function will assign the marks randomly from a valid range. Call this function for the following objects.

Student Name: Comprising two words. Each word has a length of 4-10.
Admission year: range 2020-2023.
Address: Comprise of 5 words. Each word has a length of 4-10.

Write a function (non-member) void **ShowAllAlphabetically**(Student ar[]) that displays all the student names, IDs, and GPAs according to the alphabetical order of ID.

Task 2:

The employee information system of an institution keeps records of every employee’s name, date of birth, and their respective salaries. Create a class called employee that will allow you to store all this information regarding an employee.
Write getter and setter functions for all the member variables. Before setting any value to the member variables, you need to check for these:
Name: The length has to be more than two. Otherwise, assign the default name, John Doe
Date of Birth: Every employee has an age higher than 18. If an invalid value is given, assign 1 January 2002.
Salaries: The salary has to be in between BDT 10000 and BDT 100000. If an invalid value is given, assign BDT 10000.

Define the setInfo() function, which will call all the setter functions to set the necessary information of an employee object.
Define a function named getInfo(), which will display all the stored information belonging to an employee object using the return value of the getter function.

Include one const member function named Employee& compareAge(Employee& e) which will return the elder employee object based on the date of birth.

Write an overloaded function **getAge()** by **taking system time** instead of your input for current date. You can take the help from any **reference book or internet** to check how to take system time in C++.

In the main function, create 100 employees.

Write a function (non-member) void **generateInformaiotnRandom**() takes an employee object as a parameter. This function will assign the values from a valid range. Call this function for the all created objects. Hints: See last page.

Write a function (non-member) void **ShowAllBasedOnAge**(Employee ar[]) that displays all the employees names and age, and they must be sorted in the order of age.

```
/// Some example code
#include <string>
#include <cstdlib> // For rand() and srand()
#include <ctime>   // For time()

// Function to generate random string
std::string generateRandomString(int length) {
    std::string randomString;
    const char alphabet[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    int alphabetSize = sizeof(alphabet) - 1; // Exclude null
character

    srand(static_cast<unsigned int>(time(0))); // Seed the random
number generator

    for (int i = 0; i < length; ++i) {
        randomString += alphabet[rand() % alphabetSize];
    }

    return randomString;
}

int randomInRange(int min, int max) {
    // Ensure min is less than or equal to max
    if (min > max) {
        std::swap(min, max); // Swap if min is greater than max
    }

    return rand() % (max - min + 1) + min;
}

// Function to generate a random double within a given range [min,
max]
double randomInRange(double min, double max) {
    // Ensure min is less than or equal to max
    if (min > max) {
        std::swap(min, max); // Swap if min is greater than max
    }

    // Generate a random double between 0 and 1
    double randomFraction = static_cast<double>(rand()) / RAND_MAX;

    // Scale and shift the random value to the desired range
    return min + randomFraction * (max - min);
}
```