Week 6: Operator Overloading, Use of Project feature in CodeBlocks IDE.

Learning Materials: Chapter 8

Task 1

Create a class to represent a Student information which has private data members- first name of the student, last name of the student, student id, birth year, course, and obtained marks, total number of students. Whenever a student object is created the total number of students will increase. Implement the following member functions (task of the function is written after a hyphen):

• Student(firstName, lastName, id)

• void enrollInCourses(string courseName) – students can enroll in multiple courses. Also set the initial obtained marks to 0.0

• void obtainedMarks(string courseName, float marks) – assign marks for each course.

• float setGPAForEachCourse()- returns the gpa of the course.(You can follow the traditional grading system or make your own)

• float displayCGPA()- calculated the CGPA from GPA.

• void willGraduate() – prints whether the student will graduate or not with the current marks

• void applyForScholarship() – If the CGPA is higher than 3.8, then students can apply for scholarship.

• void participateInInternship(string company) – If the CGPA is higher than 3.0 and the student has taken "X" course, then s/he can do an internship at "Y" company.

• ~ Student () – prints student full name, id, email,courses, graduation status, scholarship status and internship status.

Take input for at least 3 students with minimum 3 courses and implement in a way that all the functions have been used and also return the average CGPA of the 3 students. (If you need to use any private member, get the values using a function). Implement static and const member functions where needed.

Task 2

Create a Medicine class which has private data members - name, genericName, discountPercent, unitPrice, number of items. The default price will be 0 and discount rate - 5%. An object of the medicine class will have a unit price which is the maximum retail price. At any time a medicine can have a 0-45 % discount. It may need other member data for following functions. Implement the following member functions:

• Medicine ()

• Medicine (name, generic name, unit price)

• double updatedPrice(int percent) – return the updated price after applying a discount.

• double getSellingPrice(int nos) - this member function returns the selling price of the medicine for given nos of unit price . Selling price = price - discount.

• double readjustedPrice()- after giving a discount readjust the price for remaining medicines, so that the loss is recovered. Example: you have initially 10 items worth 10tk each. Total will be 100tk. After 1% discount you have sold 5 items for 45tk. Now you have to sell the remaining 5 items for 55tk. Calculate the unit price of them (price of one item).

• void resetPrice(): reset to initial price and display the price

• ~ Medicine () - displays the information of a medicine object in the console.

Take input for at least 3 objects and implement in a way that all the functions have been used and also return the total price of the sold medicines. (If you need to use any private member, get the values using a function). Implement static and const member functions where needed.

Task 3

Create a class "**Coordinate**". An object of the **Coordinate** class stores the abscissa and ordinate (float type). Implement the following **public** member functions (task of the function is written after a hyphen):

- **Necessary constructor, destructor and display function.**
- **float getDistance(Coordinate c)** - Distance from object c
- **float getDistance()** - Distance from origin (0,0) coordinate

- **void move_x(float val)** - val will be added to member data abscissa
- **void move_y(float val)** - val will be added to member data ordinate
- **void move(float val) -** move_x(val) and move_y(val) will be called

Write necessary member or non member functions to achieve following functionalities.

- Assume c1,c2,c3 are Coordinate objects. Overload the following comparison operators <,==,!= where distance from the origin of each operand will be compared. Example c1 == c2 returns true when c1 contains (abscissa = 1, ordinate = 1) and c2 contains (abscissa = -1, ordinate = -1)
- Unary operator ++ will move a coordinate object 1 unit in x and y direction. Implement prefix and postfix according to the convention.