NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES
ISLAMABAD CAMPUS
COMPUTER PROGRAMMING (CS103) - FALL 2018
ASSIGNMENT-3

Due Date: October 15, 2018 (05:00 pm)

Instructions:

1. *Make sure that you read and understand each and every instruction.*
2. *Plagiarism is strongly forbidden and will be very strongly punished. If we find that you have copied from someone else or someone else has copied from you (with or without your knowledge) both of you will be punished. You will be awarded straight zero in this assignment or all assignments.*
3. *Submit a single '.zip' file for your assignment and each problem solution must be provided in a folder and three separate CPP file. For instance, you must name the folder containing solution of first problem as 'q1' and second as 'q2' and so on.*
4. *You have to write each class in header and .cpp files Name the files as per your question. For instance, folder named as q1 will have three files. q1.h, q1.cpp and q1main.cpp.No code will be accepted without these files.*
5. *Note: You have to follow the submission instructions to the letter. Failing to do so can get a zero in assignment.*

Q1) Number Array Class: Design a class that has an array of floating-point numbers. The constructor should accept an integer argument and dynamically allocate the array to hold that many numbers. The destructor should free the memory held by the array. In addition, there should be member functions to perform the following operations:

- Store a number in any element of the array
- Retrieve a number from any element of the array
- Return the highest value stored in the array
- Return the lowest value stored in the array
- Return the average of all the numbers stored in the array

Q2) Implementation of Array Class: Your goal is to implement a generic “Array" class using cstring. Please also write down the test code to drive your class implementation. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

class Array{
// think about the private data members...
public:
// provide definitions of following functions...

- Array();// a default constructor
- Array(**int** size);// a parametrized constructor initializing an Array of predefined size
- Array(**int** *arr, **int** size);// initializes the Array with an existing Array
- Array(**const** Array &);// copy constructor
- **int** getAt(**int** i);// returns the integer at index [i]
- **void** setAt(**int** i, **int** val);// set the value at index [i]
- Array subArr(**int** pos, **int** siz);// returns a sub-Array of size siz starting from location 'pos'

- Array subArr(int pos);// returns a sub-Array from the given position to the end.
- int * subArrPointer(int pos, int siz);// returns an array of size siz starting from location 'pos'
- int * subArrPointer(int pos);// returns an array from the given position to the end.
- void push_back(int a);// adds an element to the end of the array
- int pop_back();// removes and returns the last element of the array
- int insert(int idx, int val);// inserts the value val at idx. Returns 1 for a successful insertion and -1 if idx does not exists or is invalid. Shift the elements after idx to the right.
- int erase(int idx, int val);// erases the value val at idx. Returns 1 for a successful deletion and -1 if idx does not exists or is invalid. Shift the elements after idx to the left.
- void size();
- int length();// returns the size of the Array
- void clear();//clears the contents of the Array
- int value(int idx);//returns the value at idx
- void assign(int idx, int val);//assigns the value val to the element at index idx
- void copy(const Array& Arr);// Copy the passed Array
- void copy(const int * arr, int siz);// copy the passed array
- void display();// displays the Array
- bool isEmpty();// returns true if the Array is empty
- Array find(int);// returns an Array containing all the indexes of integer being searched
- bool equal(Array);// should return true if both Arrays are same
- int sort();// sorts the Array. Returns true if the array is already sorted
- void reverse():// reverses the contents of the array
- ~Array();// destructor...
  }

Q3) Trivia Game: In this programming challenge you will create a simple trivia game for two players. The program will work like this:

- Starting with player 1, each player gets a turn at answering five trivia questions. (There are a total of 10 questions.) When a question is displayed, four possible answers are also displayed. Only one of the answers is correct, and if the player selects the correct answer he or she earns a point.
- After answers have been selected for all of the questions, the program displays the number of points earned by each player and declares the player with the highest number of points the winner.

In this program you will design a Question class to hold the data for a trivia question. The Question class should have member variables for the following data:

- ➢ A trivia question
- ➢ Possible answer #1
- ➢ Possible answer #2
- ➢ Possible answer #3
- ➢ Possible answer #4
- ➢ The number of the correct answer (1, 2, 3, or 4)

The Question class should have appropriate constructor(s), accessor, and mutator functions.
The program should create an array of 10 Question objects, one for each trivia question. Make up your own trivia questions on the subject or subjects of your choice for the objects.

Q4) Rational Class: Create a class called Rational for performing arithmetic with fractions. Write a program to test your class.
Use integer variables to represent the private data of the class

- ➢ Numerator
- ➢ Denominator.

Provide a constructor that enables an object of this class to be initialized when it's declared. The constructor should contain default values in case no initializers are provided and should store the fraction in reduced form. For example, the fraction 2/4 would be stored in the object as 2 in the numerator and 4 in the denominator. Provide public member functions that perform each of the following tasks:

- Adding two Rational numbers. The result should be stored in reduced form.
- Subtracting two Rational numbers. The result should be stored in reduced form.
- Multiplying two Rational numbers. The result should be stored in reduced form.
- Dividing two Rational numbers. The result should be stored in reduced form.
- Printing Rational numbers in the form a/b, where a is the numerator and b is the denominator.
- Printing Rational numbers in floating-point format.

Q5) HugeInteger Class: Create a class HugeInteger that uses a 40-element array of digits to store integers as large as 40 digits each. Provide member functions: Input, Output, Add and Subtract.
For comparing HugeInteger objects, provide functions:

- isEqualTo
- isNotEqualTo
- isGreaterThan
- isLessThan
- isGreaterThanOrEqualTo
- isLessThanOrEqualTo
- isZero

Each of these is a "predicate" function that simply returns true if the relationship holds between the two HugeIntegers and returns false if the relationship does not hold.

Q6) Car Instrument Simulator: For this you will design a set of classes that work together to simulate a car's fuel gauge and odometer. The classes you will design are:

- **The FuelGauge Class:** This class will simulate a fuel gauge. Its responsibilities are
    - ➢ To know the car's current amount of fuel, in gallons.
    - ➢ To report the car's current amount of fuel, in gallons.
    - ➢ To be able to increment the amount of fuel by 1 gallon. This simulates putting fuel in the car. (The car can hold a maximum of 15 gallons.)
    - ➢ To be able to decrement the amount of fuel by 1 gallon, if the amount of fuel is greater than 0 gallons. This simulates burning fuel as the car runs.

- **The Odometer Class:** This class will simulate the car s odometer. Its responsibilities are:
    - To know the car s current mileage.
    - To report the car s current mileage.
    - To be able to increment the current mileage by 1 mile. The maximum mileage the odometer can store is 999,999 miles. When this amount is exceeded, the odometer resets the current mileage to 0.
    - To be able to work with a FuelGauge object. It should decrease the FuelGauge object s current amount of fuel by 1 gallon for every 24 miles traveled. (The car's fuel economy is 24 miles per gallon.)

Demonstrate the classes by creating instances of each. Simulate filling the car up with fuel, and then run a loop that increments the odometer until the car runs out of fuel. During each loop iteration, print the car s current mileage and amount of fuel.

Q7) Parking Ticket Simulator: For this you will design a set of classes that work together to simulate a police of car issuing a parking ticket. The classes you should design are:

- **The ParkedCar Class:** This class should simulate a parked car. The class responsibilities are:
    - To know the car's make, model, color, license number, and the number of minutes that the car has been parked

- **The ParkingMeter Class:** This class should simulate a parking meter. The class only responsibility is:
    - To know the number of minutes of parking time that has been purchased

- **The ParkingTicket Class:** This class should simulate a parking ticket. The class responsibilities are:
    - To report the make, model, color, and license number of the illegally parked car
    - To report the amount of the fine, which is $25 for the first hour or part of an hour that the car is illegally parked, plus $10 for every additional hour or part of an hour that the car is illegally parked
    - To report the name and badge number of the police of car issuing the ticket

- **The PoliceOfficer Class:** This class should simulate a police officer inspecting parked cars. The class responsibilities are:
    - To know the police officer's name and badge number
    - To examine a ParkedCar object and a ParkingMeter object, and determine whether the car's time has expired
    - To issue a parking ticket (generate a ParkingTicket object) if the car's time has expired

Write a program that demonstrates how these classes collaborate.