NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS
Object Oriented Programming – Spring 2024

ASSIGNMENT- 4

Due Date: Wednesday 24 April 2024
Time: 11:59 pm

Please follow the following submission instructions. Failure to submit according to the format would result in a deduction of marks. Submissions other than Google classroom (e.g., email etc.) will not be accepted. No late submission will be accepted. Correct and timely submission of the assignment is the responsibility of every student; hence no relaxation will be given to anyone.

Instructions: **Total Marks: 110**

1. Make sure that you read and understand each and every instruction. **Question understanding is part of your assignment.**
2. The student is solely responsible for checking the final .cpp files for issues like corrupt files, viruses in the file, or mistakenly exe sent. If we cannot download the file from Google Classroom, it will lead to zero marks in the assignment.
3. The displayed output should be well-mannered and well presented. Use appropriate comments and indentation in your source code.
4. If there is a syntax error in the code, zero marks will be awarded in that part of the assignment.
5. Displayed output should be well mannered and well presented. Use appropriate comments and indentation in your source code.
6. **You must write driver code in main function for each question consisting of every single operator overloaded in your class. Absence of a driver code may result in deduction of marks.**

Keep a backup of your work always, that will be helpful in preventing any mishap and avoid last hour submissions. Submit a single '.zip' file for your assignment and each problem solution must be provided in a separate CPP file.

Notice: Please refrain from submitting the Dataset provided in the zip file labeled "Car details v3.csv". Failure to comply will result in a deduction of 10% of your marks.

Honor Policy

Plagiarism is a grave academic offense that can severely undermine your academic integrity and reputation. Any instance of a student found to have plagiarized their assignment, whether from a peer or external source, will be subject to strict consequences. This may result in a zero score for the current or all assignments, or in severe cases, even a failure in the course. Furthermore, all instances of plagiarism will be promptly reported to the Disciplinary Committee for further action.

Submission Guidelines

1. Dear students, we will be using auto-grading tools, so failure to submit according to the below format would result in zero marks in the relevant evaluation instrument.
2. For each question in your assignment, write your code in files provided for question 1, question 2 and so on.
3. Combine all your work in one folder. The folder must contain only .cpp files (no binaries, no exe, no .csv files etc.) Failure to comply will result in a deduction of your marks.
4. Submit the .zip file on Google Classroom within the deadline.
5. Rename the Folder as : ROLL-NUM_PROGRAM_SECTION (e.g.: i230001_AI/DS_B) and compress the folder as a zip file. (e.g., i230001_AI_B.zip). Strictly follow this naming convention, otherwise marks will be deducted.
6. The student is solely responsible to check the final zip files for issues like corrupt file, virus in the file, mistakenly exe sent. If we cannot download the file from Google classroom due to any reason it will lead to zero marks in the assignment.

Note: Start early so that you can finish it on time.

Instructions Regarding the Zip

Contents of Zip File:

- Contains dataset required for Question 1.
- Includes sample code to read CSV files.
- In the zip file from q1.cpp to q5.cpp, you'll find the operators that need to be overloaded along with their respective prototypes.

Tasks:

- Open the zip file and review its contents. For each question (q1.cpp through q5.cpp): Update and modify the provided code to overload the specified operator as per the given prototypes. Ensure that the overloaded operator class complies with the prototypes provided in the respective question file. Your implementation will be graded based on compliance with the provided prototypes. Ensure that the overloaded operators perform the specified operations accurately. Pay attention to data handling and processing, especially with the provided dataset.

Note: Please follow the instructions.

Built in functions are not allowed.

Question no 01: Marks: 30

Welcome, aspiring data scientists! In this assignment, you'll embark on an exciting journey delving into automotive analytics using C++. Your task? To unlock valuable insights from a dataset packed with detailed information about cars. From model and mileage to selling price and more, this dataset offers a treasure trove of data, providing a window into the vibrant automotive industry. Armed with only your C++ skills and a passion for analysis.

This dataset contains information about used cars. The columns in the given dataset are as follows:

1. name
2. year
3. selling_price
4. km_driven
5. fuel
6. seller_type
7. transmission
8. Owner
9. Mileage
10. Engine
11. Max Power
12. Torque
13. Seats

Your Task:

Your mission is clear: read car details from a CSV file and store them in the first garage. For the second garage, make changes manually to the same dataset, enabling testing in the main function. Remember, your tools are limited to structures, and libraries beyond reading and writing operations are off-limits. So, gear up, ignite your analytical engines, and let's dive into the thrilling realm of car data analysis. Example to read a csv file is already provided inside the zip file. (If changes are made using Rand() Function you'll get bonus marks)

Let's fire up those engines and hit the road to discovery!

Prototype:

Prototype is given already in Q1.cpp look for it.

Question no 02: Marks 20

Exploring the Power of Binary Operations

Objective:

Welcome to the realm of binary mastery! In this assignment, you'll delve deep into the mesmerizing world of binary arithmetic using C++. Equipped with the Binary class, you'll unlock the secrets of binary manipulation and unleash the power of various binary operators.

Your Task:

Implement the Binary class in C++ with the following functionalities:

1. Constructor: Accepts a binary string and a base (default value is 2). If the base is greater than 2 Convert the binary string to a base-10 integer, perform the operation, and then convert it back to a binary string with the base set to 2.
2. Overload operators for basic arithmetic operations:
    - Addition (`+`)
    - Subtraction (`-`)
    - Multiplication (`*`)
    - Division (`/`)
    - Modulus (`%`)
3. Overload bitwise operators:
    - OR (`|`)
    - XOR (`^`)
    - AND (`&`)
    - 2's Complement (`!`)
    - 1's Complement (`~`)
    - Left Shift (`<<`)
    - Right Shift (`>>`)
4. Implement input (`>>`) and output (`<<`) operators to facilitate interaction with Binary objects.
5. Ensure that the output operator (`<<`) displays both the binary and integer values of the Binary object.

6. Write comprehensive test cases in the `main` function to validate each operator overload and demonstrate their functionalities with sample inputs and outputs.

Prototype:

Prototype is given already in Q2.cpp look for it.

Question No 3: Marks 20

Objective:

- In the field of data science and AI, the term "big data" often complicates simpler tasks. Big data refers to a large volume of data that cannot be easily preprocessed or utilized. While there exist numerous algorithms in more computed languages like Python to handle such data, in C++, a language more rooted in lower-level operations, there aren't as many readily available algorithms. Consequently, we rely on our data science and AI undergraduates to develop C++ models capable of handling big data.
- The Big Float class serves the purpose of facilitating mathematical operations involving very large floating-point numbers. A floating-point number is a number expressed in the form a.b, where both a and b are integers.

BigFloat Class Requirements:

We'll implement this functionality across one file: BigFloat.cpp.

1. Functionality: The BigFloat class should handle large floating-point numbers and their arithmetic correctly.
2. Error Handling: Division and modulus operations should throw an exception if the divisor is zero.
3. Input and Output: Input and output operators should handle numbers in the form a.b or -a.b.
4. Modulus Operator: The modulus operator (%) should return a BigFloat representing the remainder of the division of the current BigFloat by the given BigFloat. For example, if a = 10.75 and b = 3.25, then a % b should be 1.0.
5. Precision Control: The precision of the BigFloat should be controlled by the precision data member. This data member should determine the number of digits after the decimal point for all calculations and outputs. For example, if precision = 2, then the BigFloat 1.2345 should be represented as 1.23.

Prototype:

Prototype is given already in Q3.cpp look for it.

Question No 4: **Marks 20**

Introduction:

Sets in mathematics are fundamental collections of distinct elements without specific order or repetition. They form the basis of various mathematical disciplines. Sets adhere to the principle of extensionality, where two sets are equal if they contain the same elements. Operations on sets include union, intersection, and complementation, altering elements to create new sets. Set theory explores advanced concepts like cardinality and transfinite numbers, providing a framework for mathematical reasoning. Sets categorize and define mathematical objects, aiding in formalizing mathematical systems. They are essential for expressing mathematical concepts and relationships concisely. Sets serve as the foundation for structures like groups, rings, and fields, enabling the study of algebraic systems. In mathematical discourse, sets facilitate communication and dissemination of mathematical knowledge across disciplines. Overall, sets are vital tools in mathematics, shaping the understanding and exploration of mathematical theories and structures.

Your Task

We'll implement this functionality across one file: q4.cpp.

Your goal is to overload the operators for a generic "Set" class. The Set class should handle sets and their operations correctly.

1. The union operation (+) should return a Set that contains all elements that are in either set.
2. The intersection operation (*) should return a Set that contains all elements that are in both sets.
3. The difference operation (-) should return a Set that contains all elements that are in the first set but not in the second set.
4. The square bracket operator ([]) should return a reference to the element at the given index in the set.
5. The find function should return the index of the given element in the set, or -1 if the element is not found. The sort function should sort the elements in the set in ascending order.
6. The input and output operators should handle sets in the form {a, b, c, d}, where a, b, c, and d are elements of the set.

All the instructions are properly mentioned with the function's prototype read them Carefully.

Prototype:

Prototype is given already in Q4.cpp look for it.

Question No 5: **Marks 20**

Introduction:

The Time class is a versatile tool for managing time representations, whether it's for the time of day or durations. Here, we'll implement a function to calculate the elapsed time since a given Time object. This function will return a new Time object representing the amount of time that has passed since the given instance.

Instructions:

To achieve this, we need to overload operators for a generic "Time" class. We'll implement this functionality across one file: Time.cpp.

Implementation Steps:

1. Define Class and Data Members: Define a class named Time with three integer data members to store hours, minutes, and seconds.
2. Constructor and Copy Constructor: Implement a constructor to initialize time values. Implement a copy constructor.
3. Binary Operators: Overload arithmetic operators + and -.
4. Compound Assignment Operators: Overload compound assignment operators += and -= to perform addition and subtraction.
5. Logical Operators: Overload logical operators to compare time instances.
6. Additional Functions: Implement elapsedTime() to calculate elapsed time since a given instance.
7. Destructor: Define a destructor to release resources if necessary.
8. Input and Output Operators: Overload << and >> operators for input and output operations.
9. Testing: After implementation, test the class with various scenarios to ensure correctness and robustness.

Prototype:

Prototype is given already in Q5.cpp look for it.

***** Good Luck *****