National University of Computer and Emerging Sciences, Lahore Campus

Course:	Object Oriented Programming	Course Code:	CS1004
Program:	BS (Software Engineering)	Semester:	Spring 2023
Due Date:	30 March 2023	Total Marks:	60
Section:	SE(2C)	Weight	
Evaluation:	ASSIGNMENT#2	Page(s):	2

Submission Path: Google classroom

Instructions:

- Do not add any extra variable(s) to the class
- Submit only one **RUNNING** file as YourRollNumber.cpp that contains class, its implementation and the driver Program. **Do not submit .rar or .zip file**. Plagiarism policy already conveyed.

QUESTION 1:

You are required to implement the following class in your assignment

```
Class VectorType
{
private:
//necessary variables of pointers type
    double *x;
    double *y;
    double *z;

public:
//Consider U and V are two objects/Vectors of type VectorType. You
are required to implement the following operations
```
1. **[1]** Default and parameterized constructor
2. **[1]** Copy Constructor
3. **[1]** Destructor to de-allocate dynamic memories
4. **[1]** Overloaded assignment operator
5. **[2]** Function to find the dot product of two vectors
   a. For this you will need to overload * operator
      Dot product between U and V can be determine by the following mathematical formula
      $$U * V = (U.x * V.x) + (U.y * V.y) + (U.z * U.y)$$
6. **[1]** Function to find length of a vector
   a. Length V can be determined by the following mathematical formula.
      $$LenV = \sqrt{x^2 + y^2 + z^2}$$

7. **[3]** Function to find angle between two vectors V and U
   a. Angle can be determined by using the following formula

$$\theta = cos^{-1}(\frac{U*V}{Len\ U*Len\ V})$$

8. **[2]** Addition and subtraction overloaded operators for vectors
9. **[4]** Overload pre and post increment and decrement operators (++ and --)
10. **[4]** Overloaded ==,!=, >> and << operator as **non-member** function(**friend**).

};

QUESTION 2:

String

Implement a custom type for representing strings in C++. Allow following operations as well as

operators, considering dynamic memory allocation:

Operations:

• length: determine the length of string

• upper: convert the string to upper case

• lower: convert the string to lower case

• at: return character at a given index

• substring: extract a substring given start and end

• index: find starting index of a substring

• compare: compare two strings

• concat: concatenate/append the argument after current string. Cater cases for different data

types such as String, C-string, char, int, float

• prepend: concatenate/append the argument before current string. Cater cases for different data

types such as String, char, int, float

Operators:

• + : for concatenation and prepend operations taking into account different data types and order of arguments.

- = : for assignment

- ==, !=, < and > : for comparison operations

- [] : for access to character at a given index

- >> and << : for output and input a string

QUESTION 3:

Roman Number

Implement a class to represent a Roman Number based upon the
Standard Form described on the
Wikipedia page (https://en.wikipedia.org/wiki/Roman_numerals).
Support following operations using
corresponding operators:
- + : for adding two roman numbers
- - : for subtracting two roman numbers
- * : for multiplication of two roman numbers
- / : for division of two roman numbers
- ==, !=, < and > : for relational comparison of two roman numbers
- ++ and - - : for increment and decrement, both prefix and postfix
versions