University of Central Punjab

(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)

FACULTY OF INFORMATION TECHNOLOGY

Fall 2020
CSCP2023 -Object Oriented Programming
Assignment # 02

Deadline: 22/11/2020 Sunday, 11:59 PM **Total Marks:** 50

Important Note:

This is an individual assignment, working in groups or taking help from your peers will be considered as plagiarism. Understanding of the question is part of the question nevertheless you can ask your teacher only to clarify about what is required in the question. Copying a single line of code from internet sources or from your peer's solution can award you a straight ZERO in whole assignment or even worse.

Question 1: (10 points)

This question requires you to write code to manipulate **Date**. Design your application using the following class description:

```
class Date {
public:
        Date();                 // initialize attributes with default values (01-01-2000)
        Date(int,int,int);  // if invalid, set members equal to default values (01-01-2000)
        Date(const Date&);      // A regular copy constructor
        bool inputDate();       // assign values to date attributes (cin from user),
                                     // return true if valid range of values are input else false
        bool copyDate(Date&); // will behave similar to copy constructor
        bool inputCompleteDate(int,int,int);    // will behave similar to parameterized constructor
        Date& getDate() const;
        void retrieveDate(int& , int& , int&) const; //retrieve Date attribute values using "pass by reference"
        void showDate() const; // Display Date attribute values on Console
        bool isEqual(Date&);   // return true if calling & passing date objects are equal
        bool isLeapYear();
        ~Date();        // A regular destructor
        // All other relevant methods like setter/getter etc also to add
private:
        bool validateDate(); // Validate attribute range values
        int day;
        int month;
        int year;
};
```

Important Note: Make attributes, functions, parameters and objects as CONST wherever possible. It carries marks.

Question 2: (10 points)

Repeat the same problem as in Question 1 but for "**Time**". Use HH:MM:SS format for time. Implement all functions as described in that question.

Also give support for both formats (12-hour & 24-hour), to support 12-hour format add an attribute of type char* for "AM" or "PM". In case of 24-hour, this attribute will contain nullptr. Provide functions to convert from 12-hour to 24-hour and vice versa. Change all methods accordingly. (Ignore isLeapYear() function for this task)

Important Note: Make attributes, functions, parameters and objects as CONST wherever possible. It carries marks.

Question 3: (10 points)

Part 01

Make a class **Name** with implementation of all functions mentioned and check all your functions at least once in the main program. The description of class name is given in the header file below:

```
class Name {
public:
        Name(char* = nullptr, char* = nullptr); // parameterized constructor with default arguments
        Name(const Name&);             //copy constructor
        ~Name();                        //destructor (should release all dynamic allocations)
    void copyName(Name&); // this is not a copy constructor, I want this function to copy the contents of
                          // one name(firstName and lastName both) to another name.
        void camelCase(); // make first letter capital of both attributes
        void toLower(); //convert name to lower case alphabets
        void toUpper(); //convert name into upper case alphabets
        int nameLength(); // both first and last (excluding space)
        void swapNames(); // firstName becomes lastName and vice versa
        void display(); //prints name(firstName and lastName with space in between)
        char* fullName(); //concatenate both attributes and return full name with a space in between both
        // provide setters/getters for Name class. Remember: Setters with no memory leakage.
        // provide other relevant methods you want like your own strLength and strCopy function etc.
private:
        char* firstName;
        char* lastName;
        bool isValidName();// name should contain only alphabets - no special characters or digits
};
```

Implementation is to be done in multiple files. Your class should also be responsible for input validation.

Part 02

You have to provide a **NameCompare** function which is not the member of class name. (Can be written in source.cpp in which you have written main function).

Function Signature: int NameCompare(Name name1, Name name2)

This function takes two names as arguments and compare these two names, first you have to compare lastName (lexicographically) and then firstName (lexicographically)

Hint: Compare only if the result of matching Last Names is returned as **matched**.

- **NameCompare**() compares the **two names lexicographically** means it starts comparison character by character starting from the first character until the characters in both names are equal or a NULL character is encountered.
- If first character in both names are equal, then this function will check the second character, if this is also equal then it will check the third and so on
- This process will be continued until a character in either name is NULL or the characters are unequal.

About Returning Value:

This function can return **three different integer values** based on the comparison:

- **Zero (0)**: A value equal to zero when both names are found to be identical. That is, all of the characters in both names are same.
- **Greater than zero (>0)**: A value greater than zero is returned when the first not matching character in name1 have the greater ASCII value than the corresponding character in name2 or we can also say If character in name1 is lexicographically **after** the character of name2.
- **Less than Zero (<0)**: A negative value is returned when the first not matching character in name1 have lesser ASCII value than the corresponding character in name2. If character in name1 is lexicographically **before** the character of name2.

Important point: When the names are not same, you will find that the value returned by the NameCompare() function is the difference between the ASCII values of first unmatched character in name1 and name2 in both the cases.

Question 4: (10 points)

Write a class **LibraryBook** with following data members to use it as part of Library Management System at a later point.

• ISBN (char * const)	// ISBN = International Standard Book Number
• title (char *)	// Title of Book
• author (const Name)	// constant Object of **Name** class implemented in Question 3
• publisher (Name)	// Object of **Name** class implemented in Question 3
• quantity (int)	// How many copies of book are available in library
• lastIssue (Date)	// Object of **Date** class implemented in Question 1

➔ Add all kind Constructors, setters, getters, destructors and related utility functions to the class.

Question 5: (10 points)

A matrix is a rectangular storage of numbers, symbols or expressions, arranged in rows and columns. Write a program to design a class to represent a matrix. Class objects should be able to store elements of matrix in a 2-D dynamic array of integers. Also class should store information about number of rows and columns.

Class data members should be like following:

```
class Matrix {
 private:
   int ** elements;
   const int rows;
   const int cols;
};
```

University of Central Punjab

(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)

FACULTY OF INFORMATION TECHNOLOGY

Add following function to the class:

Parameterized constructor with default arguments:
Default values for row and column should be 1 and nullptr for pointer. User can also initialize matrix elements with values in constructor.

Copy constructor:	You know what it does.
Destructor:	De-allocate all allocated memories properly.
void CopyMatrix(Matrix obj):	Not a constructor, but will do same what copy constructor do.
void setElement(int val, int row, int col):	A function to set a specific element in the matrix
int getElement(int row, int col):	A function to get a specific element from matrix.
int getNoOfRows():	Getter function for **rows** data member
int getNoOfCols():	Getter function for **cols** data member
int* getRow(int):	This function will return a complete row.
int* getCol(int):	This function will return a complete column.
void setRow(int*, int):	This function will set a complete row.
void setCol(int*, int):	This function will set a complete column.

setRow() and setCol() function will make sure that number of elements in passing row or column should be equal to existing size of row and column.

Matrix Transpose():	Will return transpose of the matrix in a new Matrix object.
bool isSquare():	Check if matrix is square matrix or not. Square matrix is one which have equal number of rows and columns.
bool isDiagonal():	Check if matrix is diagonal matrix or not. Diagonal matrix is one who is **square matrix** AND have value zero at all non-diagonal* places.
bool isIdentity():	Check if matrix is identity matrix or not. Identity matrix is one which is **square matrix** AND have value 1 at diagonal and 0 at all non-diagonal places.
bool CompareMatrix(Matrix one, Matrix two):	A global function (not member of class) to compare if two matrices are equal or not.

Note:

- Be very careful regarding memory leakage, sizes, dangling pointers and other memory related issues on return values while coding this class.
- Make use of CONST keyword wherever possible with data members, member functions, parameters and return values.

** Diagonal are those places in matrix where row # and column # is equal.*

Submission Procedure:

1. Submission is to be done thru UCP Portal. “Assignment 02 Submission Folder”
2. Do not submit the DEBUG folder.
3. Upload a single COMPRESSED file (extension should be only .RAR) containing a folder for each question.
4. The name of the COMPRESSED folder should be the same as your registration number.
   (Example, a student having registration number L1F19BSCS0001 will upload a single file having a name: L1F19BSCS0001.rar)