National University of Computer and Emerging Sciences

Lab Manual 05
Object Oriented Programming – CL1004

Course Instructor	MS. Anoosha
Lab Instructor(s)	Ms. Amna Zulfiqar Ms. Samman Ashraf
Section	BCS-9C
Semester	Summer 2023
Date	12-07-2023

Department of Computer Science
FAST-NU, Lahore, Pakistan

Lab Manual 05 – Class

Important Note:

- **You may find the syntax to accomplish these exercises from lecture demo.**
- **Add Necessary Comments in you code to justify your logic.**
- **Comment exercise number or statement at the start of your code**
- **Save each exercise in .cpp file with your roll no, ex and lab number e.g.**
- **22LXXXX_EX01_Lab01.cpp**
- **Upload cpp files on google classroom**
- **Make sure that the interface of your program is user friendly i.e. properly display information.**
- **Properly follow the coding standards.**

1. Exercise:

Create a class "Matrix" that represents a 2D matrix with private member rows, column, and ****data** (a pointer to a dynamically allocated 2D integer array).

Add public member functions

1. **Matrix(int rows, int cols): allocate memory for the matrix**
2. **~Matrix():** Destructor to deallocate memory for the matrix
3. **void input()** to initialize the matrix with user input
4. **void print()** to print the matrix
5. **void transpose()** to transpose the matrix.
6. **Add a Deep Copy Constructor to your MatrixClass.**

Also Add Usage Example of each function of your class in main()

2. Exercise:

Design a class **Holiday** that represents a holiday during the year. This class has three **private** data members:

- **name**: A string that represents the name of holiday.
- **day**: An integer that holds the day of the month of holiday.
- **month**: A string that holds the month the holiday is in.

1. Write a default constructor that initializes each data member of class such that **name** with **NULL**, **day** with **0** and **month** with **NULL Holiday()**

2. Write a constructor that accepts the arguments for each data member such that **string n** assigned to **name**, **int d** to **day** and **string m** to **month**. **Holiday(string &n, int d, string &m)**

Note: Use member function initialization for all data members.

3. Generate getter setter of each member variable: such that **name** should never be greater than 50 characters, **day** should never be negative and **month** should not be greater than 10 characters.
    - bool setName(string &s)
    - string getName()
    - bool setDay(int u)
    - int getDay()
    - bool setMonth(string &p)
    - string getMonth()

4. Write a function **inSameMonth** (outside class) which takes two Holiday objects as arguments, compares two objects of the class Holiday, and returns true if they have the same month otherwise false.
   bool inSameMonth (Holiday &a, Holiday &b)

5. Write a function **avgDate** (outside class) which takes an **array** of type Holiday and its **size** as its argument and returns a **double** value that is the average of the entire day data member in the Holiday array **arr**. You may assume that the array is full (i.e. does not have any NULL entries).
   double avgDate(Holiday arr[], int size)

Note: You must include Usage Example of each function of your class in main().

3. Exercise:

Design a class **ComplexNumber** for handling Complex numbers

1. Implement following **ComplexNumber** class

```
class ComplexNumber
{
private:
        int* real;        //Integer will be saved on heap
        int* imaginary;          //Data will be saved on heap
public:
        ComplexNumber(int, int); /*Constructor with Default arguments, allocate memory
properly and display Constructor Called message with data.*/
        ~ComplexNumber(); /*Deallocate memory properly and Display Destructor Called
message with data.*/
        void Input();    //Takes input from user
        void Output(); //Properly display Complex number like a+bi
        float Magnitude();      //Calculates magnitude of a complex number
};
```

2. Create a complex number c1 on stack using default constructor. Take c1's data using your input function. Display c1 using Output function.

3. Create a ComplexNumber pointer cPtr in your main and check if it implicitly calls Constructor or not.

4. Create a Complex Number 3+5i on heap and save its address in cPtr. Print the complex number using cPtr.

5. Create an array of five complex numbers on stack. Take complex numbers input from user in for loop. Display all these complex numbers in for loop, along with their magnitude. Do we need to delete this array?

6. Dynamically create an array of complex numbers after taking the size of array from user. Input and output these complex numbers using Input Output functions. Also display their magnitude. What will be the memory configuration in this case? Do we need to de-allocate anything?