K J Somaiya Institute of Management Studies and Research
MCA Batch : 2016-19 – **Semester** : 1
Subject : Programming and Object Oriented Concepts with C++

Maximum Marks : 50 **Time** : 3 hours **Date** : 23rd Nov, 2016

Note : Question number 1 is <u>compulsory</u>. Answer **<u>any four</u>** of the remaining six questions.

Q.1. Create a class **MyDate** with the basic public interface functions like : **[14 Marks]**

- A constructor which acts as a default cum 3-arg constructor that populates the caller object. If an object does not pass arguments then, the default date will be 1-1-2000.
- setDate() will populate an object through the keyboard.
- validate() will be invoked by setDate(). It will validate the date and return 1 if valid else throw three different exceptions called dayError, monthError or yearError depending on the error. These errors must be defined in the MyDate class. Any year outside the range 1900 and 2099 is erroneous.
- print() function must print the details of the calling object in formatted output.

The client code will demonstrate the class implementation inside the try block, followed by the three catch blocks for the three errors thrown, which will display suitable error messages.

Q.2. Create a class **Arithmetic** as defined below. Now create an **array of n dynamic Arithmetic objects,** where n is accepted from the keyboard in function main(). Now populate each object via the keyboard through a getData() function. Populate only the integer value (not sumOfDigits).

Once the objects are populated, invoke another function called compute() which computes the sumOfDigits member for each object in the array. For example if the value of the object is 7865, the sumOfDigits of that object will be 26, which is the sum of 7,8,6 and 5. Finally print the array contents in a formatted output via the print() function. The output must have proper headers.

class Arithmetic **[9 Marks]**

```
{
        private :
                int  value;
                int  sumOfDigits;
        public :
                …  …  …
                …  …  …
};
```

Q.3. Write theory notes for the following **<u>each with an example :</u>** **[9 Marks]**

a) Explain inline functions in detail.
b) Differentiate between function overloading and function overriding.
c) Explain any three features of Object Oriented Programming in detail.

Q.4. Create a class **ShortDate**, which has integer day, month and year as private data. Now create one more class called **LongDate** which has an int day, char string month (length 4) and an int year as private members. Both classes must have basic functions like a default constructor, argument constructor, and print etc. **[9 Marks]**

Class ShorDate must first validate the date received from the client. Only if valid can the coversion process be executed. If invalid, an error message must be flashed and there should be no conversion.

Class LongDate must additionally have an **overloaded stream insertion operator function** to display an object's content in formatted output such as dd monthName yyyy. The main difference in their types being that class ShortDate holds month as an int between 1 and 12, whereas class LongDate holds month as a character string like : "Jan", Feb" … "Dec".

Write a program to convert a ShortDate object into a LongDate object using an overloaded type cast operator function. A sample client code is given below :

```
main()
{
        ShortDate SD(15, 8, 2016);
        Longdate LD = SD;    // Code conversion routine in the source class

        SD.show();           // should print  15/8/2016
        cout << LD;          // should print   15 Aug 2016
}
```

Q.5. Create a hierarchical inheritance hierarchy with Shape as a base class and class Circle, Square and Triangle inheriting from class Shape. The basic structure of each class is given below : **[9 Marks]**

```
class Shape
{
        public :
                void print() = 0;
                double area() = 0;
};
```

```
class Circle : public Shape
{
        private :
                double radius;
                public :
                …. …. ….
                …. …. ….
};
```

```
class Square : public Shape
{
        private :
                int length, breadth;
        public :
            …. …. ….
            …. …. ….
};
```

```
class Triangle : public Shape
{
        private :
                int base, altitude;
        public :
                    …. …. ….
                    …. …. ….
};
```

All the three derived classes **must override** the pure virtual print() and area() functions of the base class Shape. Apart from this, the derived classes must have constructors, and destructors printing simple messages..

Now, write a driver program that creates an array of three pointers of type Shape. This array must be initialized with addresses of three objects of each of the derived classes. In a loop invoke the area() and print() functions virtually.

Q.6. Create a **class template** for the **Stack class**. The class must have a dynamic array, top (top of stack) and its size as private data members. Include the pop(), push(), constructors and destructor functions. Include any other function you deem fit.

Write a driver program to demonstrate the parameterized class for a **stack of integers and a stack of doubles.** **[9 Marks]**

Q.7. Create a **class Array** which holds a **dynamic** integer array, its size and total, which holds the number of objects created. It should have the following methods : **[9 Marks]**

I. A default constructor which creates a dynamic array of size 5 for all objects whose size has not been passed by the client.
II. A one-argument constructor which creates a dynamic array of the size passed as the argument.
III. A copy constructor which is an object initialization with another Array object.
IV. A set function which populates a calling object with data from the keyboard.
V. An overloaded assignment operator function.
VI. A destructor function which frees the dynamic memory of each object that goes out of scope.
VII. A print function which prints all the array elements of the calling object in formatted output.
VIII. An overloaded ++ function which increments every element of the array of the calling object by 1.
IX. An overloaded == function which returns a true if two Array objects are identical or false if they're not.
X. An overloaded [ ] subscript operator function which returns the reference of the requested array element, received by the client and printed.

Write a sample driver program to demonstrate the class implementation.

****