EAST WEST UNIVERSITY
Department of Computer Science and Engineering
B.Sc. in Computer Science and Engineering Program
Final Examination, Spring 2021 Semester

Course: CSE 110 Object Oriented Programming, Section-5
Instructor: Tanni Mittra, Senior Lecturer, CSE Department
Full Marks: 35 (35 will be counted for final grading)
Time: 1 Hour and 30 Minutes
Submission Time 10 Minutes

Note: There are **6** (**Six**) questions, answer ALL of them. Course Outcome (CO), Cognitive Level and Mark of each question are mentioned at the right margin.

---

1. Create an Abstract class ***Number*** with two instance variables digits[] and base. It has two Abstract methods also ***ConvertToDec() & DecToBase(int number)***. [CO3, C3, Mark: 8]

Create a class ***BinaryNumber*** that will extend ***Number*** class. Override ***ConvertToDec()*** method in such way that will convert digits[] into corresponding decimal numbers. For this class consider that, ***ConvertToDec ( )*** will take digits[] as binary number and base as 2. Again override ***DecToBase ()*** method which will take a decimal number and convert it to binary number.

Example:

Digits[5] = 1,0,1,0,1

$(1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 21$

21 / 2 = 10	1
10 / 2 = 5	0
5 / 2 = 2	1
2 / 2 = 1	0
1 / 2 = 0	1

In reverse order 10101

Create a class ***OctalNumber*** that will extend ***Number*** class. Override ***ConvertToDec()*** method in such way that will convert digits[] into corresponding decimal numbers. For this class consider that, ***ConvertToDec ( )*** will take digits[] as octal numbers and base as 8. Again override ***DecToBase ()*** method which will take a decimal number and convert it to octal number.

Example:

Digits[3] = 2,1,5

$2 \times 8^2 + 1 \times 8^1 + 5 \times 8^0 = 141$

141 / 8 = 17	5
17 / 8 = 2	1
2 / 8 = 0	2

In reverse order 215

4 / 8= 0 4

2. Define a Generic class named ***Rectangle*** that has three instance variables: ***length*** ***height*** and ***width***. Where each of the input can take objects of primitive data type classes. The partial class diagram of ***Rectangle*** class is given below. [CO3, C3, Mark: 5]

Rectangle <T>
- length: T - width: T -height: T
+ **Rectangle** (T len, T width,T height) +DisplayResult(): void + Perimeter(): T + Area():T + volume():T +Is_square():Boolean

Complete this class definition by including all the methods. DisplayResult() method will display the result of perimeter, area, volume and result of is_square. Write a main method that create two objects of Rectangle class of Integer & Double type. Performs the Rectangle operations and displays the output.

3. Objects of the ***Rectangle*** class as defined in question 2, must have non-negative ***length*** and ***width***. If **length** is equal to 0 and **width** is positive, or vice versa, then it is a line. If both are 0, then it is a point. The Rectangle class must adhere to this rule. Create the IllegalRectangleException class, and modify the Rectangle class to throw an IllegalRectangleException object if the above mentioned rules are violated. [CO3, C3, Mark: 5]

Define a user-defined exception class IllegalRectangleException.

Then, modify the ***Rectangle*** class such that if the above rules are violated, the class must throw a IllegalRectangleException. Write only the part of the class that you have modified. No need to write the full definition again.

Write only the part of a main method within the Main class that creates a Rectangle type object, take input of two instance variables surrounded with appropriate try-catch block.

4. Create an **Account** class that has instance variables *accname*, *withdraw*, *deposite* along with appropriate constructors, accessors, and mutators(if needed). Make the **Account** class implement the Comparable interface. Define the compareTo method to order **Account** objects based on the deposited amount. In the main method, create an array of at least five **Account** objects, sort those using Arrays. Sort, and output the **Account**. They should be listed by ascending deposited amount. [CO3, C3, Mark: 6]
Next, modify the compareTo method so it orders **Account** objects based on the

lexicographic ordering of their account name. The program should now output the account ordered by name.

5. Consider the following class diagram. [CO3, C3, Mark: 6]

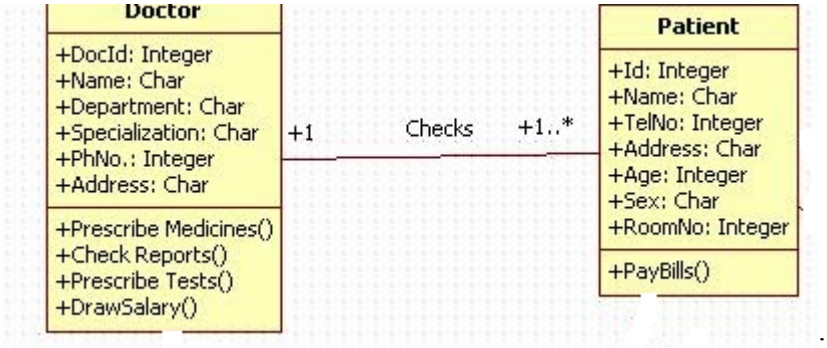

Figure 1: A class diagram

Assume that the above-mentioned classes have already been implemented. Also consider that you have a Main class and within the main () method in which the following array lists of Doctor and Patient type objects are created.

```
ArrayList<Doctor> doctors = new ArrayList<>();
ArrayList<patient> patients = new ArrayList<>();
// adding few doctors and patients
doctors.add(……);
patients.add(……);
```

a) **Complete** the above program in Java that writes all Doctor type objects as created and held in *doctors* ArrayList into a file named **"doctors.dat"** using **DataOutputStream** class. You must write the data in the following format.

```
DocId Name Specislization department phNo Address
```

b) Do you need to change the class definition of the above-mentioned classes to write objects of those classes directly into a file using **ObjectOutputStream** class? If yes, what would be the change?

6. Write a java program that will create one threads class. Where the thread class will perform add operation of two numbers. Write a main method where you have to create three threads of that class and each thread will perform two add operations. You have to ensure that each thread will complete their operations then next thread will begin to run. [CO3, C3, Mark: 5]