B.Sc. (Honours) Examination 2018
Semester-IV
Computer Science
Course : BCSC-41
(Object-Oriented Programming in Java)

Time : 3 Hours **Full Marks : 40**

Questions are of value as indicated in the margin

Answer Question No. **1** and **any three** from the rest

1. Answer **any five** questions : 2×5=10
   a) Can you override a private or static method in Java? State the reasons.
   b) Can we cast any other type to Boolean Type using type casting?
   c) State the differences between Checked Exception and unchecked Exception.
   d) Can there be any abstract method without an abstract class?
   e) Is it possible to import a class or a package twice? Will the JVM load the package or class twice at runtime?
   f) How string objects are different from string buffer objectrs?
   g) State the purpose of toString () method in JAVA?
   h) With example, state the purpose of finalize() method.
2. Is it necessary to follow each try block by a catch block? With example, explain why sometimes we place multiple catch blocks after a try block. With examples, explain Constructor Overloading and Constructor Chaining. Which class is the supreclass for every class? 2+3+4+1=10
3. What is the difference between inner class and nested class? In Java, when should we use an interface instead of an abstract class? Can you declare an interface method as static? Discuss different types of access level modifiers and their uses. 2+2+2+4=10
4. Let $d(n)$ be defined as the sum of proper divisors n (number less than n which divide evenly into n). If $d(a) = b$ and $d(b) = a$ where $a \neq b$, then a and b are amicable pair, and each a and b are called amicable numbers. For example, the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, and 110; therefore d (220) = 284. The proper divisors of 284 are 1, 2, 4, 71, and 142; so d (284) = 220.
   Write a JAVA program to evaluate the sum of all the amicable numbers under 10000.10
5. Working from left-to-right if no digit is exceeded by the digit to its left it is called an *increasing* number; for example, 134468. Similarly, if no digit is exceeded by the digit to its right it is called a *decreasing* number; for example, 66420. We shall call a positive integer that is neither increasing nor decreasing a *bouncy* number; for example, 155349.
   Clearly, there cannot be any *bouncy* numbers below one-hundred, but just over half of the numbers below one-thousand (525) are *bouncy*. In fact, the least number for which the proportion of bouncy numbers first reaches 50% is 538. Surprisingly, bouncy numbers become more and more common and by the time we reach 21780 the proportion of *bouncy* numbers is equal to 90%.

P.T.O.

Write a JAVA program to find the least number for which the proportion of *bouncy* numbers is exactly 99%. 10

6. Each new term in the Fibonacci sequence is generated by adding the previous two terms. Starting with 1 and 2, the first 10 terms will be : 1, 2, 3, 5, 8, 13, 21, 34, 55, 89.

   Every positive integer can be uniquely written as a sum of nonconsecutive terms of the Fibonacci sequence for example 100 = 3+ 8 +89. Such a sum is called the *Zeckendorf* representration of the number.

   For any integer n > 0, let z (n) be the number of terms in the *Zeckendorf* representation of *n*. Thus z(5) = 1, z(14) = 2, z(100) = 3 etc.

   Write a JAVA program to find the value of $\sum_{n=1}^{1000} z(n)$. 10

———