1.	**FizzBuzz** Time: 25 Minutes Write a method named *getFizzyBuzz* in a class *FizzBuzz*. The method should take an integer *n* as a parameter and return a string. The logic should be: 1. If n is divisible by 3, return *“Fizz”*. 2. If n is divisible by 5, return *“Buzz”*. 3. If n is divisible by both, return *“Fizzbuzz”*. 4. Otherwise, return *“Boom”*. Write at least 4 unit tests to validate each of the cases.	5
2.	**MaxStack** Time: 50 minutes You need to create a class named *MaxStack* that represents a last-in-first-out (LIFO) data structure with the following properties: 1. It has *push(int)* and *pop()* operations that work the same way as a normal stack. 2. In addition, it has a *max()* operation that returns the maximum value in the current stack. **Constraints** The *max()* operation should operate at constant complexity, *O(1)*. This means you cannot use a loop or recursion to find the minimum value. **Test cases** 1. Push 3, 2, 5, 1. Assert max = 5. 2. Push 3, 2, 5, 1. Pop. Assert max = 5. 3. Push 1, 2, 3, 4. Assert max = 4. 4. Pop 4 from 1, 3, 4, Assert max = 3. **Hint** 1. You can use the built-in Stack class if necessary. 2. An additional hint will be given after 15 minutes if you ask for it.	10
3.	**Practice Generics (Implement the concept discussed in the Theory class)** Time: 35 minutes You have to create a generic method printList in Printer class that takes a list of Faculties or students i.e., can support an Inheritance hierarchy. 1. Create a Person class having a name, address, and age attribute. Override toString method.	5

	2. Create a Student class that extends the Person class and has one additional attribute studentID. Override toString method. 3. Create a Faculty class that extends the Person class and has one additional attribute designation. Override toString method. 4. Create a generic Printer class and declare a generic method.	
	Bonus Generics Convert the task 2 into a generic implementation [If you can complete this you will get 5 marks bonus.]	