Java Basics Assignment Part 1 Total Marks : 120

Note : First nine questions are of 10 marks, last question is of 30 marks

Q1. What will happen if the hashCode function is overridden in this way :

```
@Override
public int hashCode() {
   return rand.nextInt(1000);  // rand.nextInt() finds a random number
}
```

Q2. What will happen if we have a HashMap in java like this and we have not overridden equals function in the Customer.java class

```
HashMap<Customer, Integer> hashMap = new HashMap<>() // Customer is a class
defined in java by you let's say
```

Q3. What will happen if we don't write @Override annotation over equals or hashCode function in any class (let's say Customer.java). Can we still override the function, if no then why no, if yes then what's the purpose @Override is solving

Q4. In Java streams
a) can we write a filter expression after forEach expression
b) can we write a filter expression after map expression

Explain can we perform a and b. If yes then why yes, if no then why no

Q5. Is this piece of code correct, write an explanation to support your answer?

```
public interface questions{

        default void print() {
                System.out.println("Hello");
        }
}
```

Q6. Is this piece of Java Streams code correct, if yes then explain why and if not then write the correct one ?

```
numbers.map(x -> x*x).forEach(System.out::println)
```

Q7. Write a program to print 5 random numbers using Java streams?

Q8. Write a program using Java Streams to square the list of numbers and then filter out the numbers greater than 100 and then find the average of the filtered numbers

Q9. Write a program using Java Streams to find the lowest and highest number in a list of integers

Q10. Write a program in Java to store the information of a customer in Map in a sorted fashion of the key

```
For example :

1 -> {“custId”: 1, “name” : “Ram”, “age”: 20, “gender”: “male”}
2 -> {“custId”: 2, “name”: “Rashmi”, “age”: 40, “gender”,
“female”}
3 -> {“custId”: 3, “name”: “Kiran”, “age”: 30, “gender”: “female”}
4 -> {“custId”: 4, “name” : “Karan”, “age”: 25, “gender”: “male”}

…………………….

Here key is the customer id on the basis of which we have to sort.
Currently the Customer class is something like this

Customer{
      int custId;
      String name;
      String gender;
      int age;
}

numbers.map(x -> x*x).forEach(System.out::println)

The marking will depend on the complexity of operations

You have to write 3 functions :

Insert(custId) - insert a customer in a sorted fashion
delete(custId) - deletes a person from map and the map still
remains sorted
find(custId) - finds a person in the map

print() – finds all the customer in the map in a sorted fashion
```