4. What is the output of the following program?

```
String s1 = "Java World";
String s2 = new String("Java World");
String s3 = "Java World";
System.out.println("s1 == s2 is " + (s1 == s2));
System.out.println("s1 == s3 is " + (s1 == s3));
```

A. s1 == s2 is false
s1 == s3 is true

B. s1 == s2 is true
s1 == s3 is true

c. s1 == s2 is false
s1 == s3 is false

D. Error

5. What is the output of this given code?

```
public class Test {
    static int count=0;
        int tot=0;
    Test(){ ++count;
            ++tot;
          }
    public int getCount(){return count;}
    public int getTot(){return tot;}

    public static void main(String[] args){
      Test instance = null;
      for(int i = 0; i < 5; ++i){
            instance = new Test();
      }
      System.out.println(instance.getCount() +"\t" + instance.getTot());
    }
}
```

A. 1 1

B. 5 5

C. 5 1

D. 1 5

6. Which statement below is true when the following code is compiled/run, and the input is a non-null `String start` and a `char c`?

a. Invalid recursion, which leads to a runtime error.
b. Outputs true if the input string is empty, false otherwise
c. Outputs true if the character c is contained in the string start, false otherwise
d. Outputs true if the character c is *not* contained in the string start, false otherwise

```
public static boolean what(String start, char c){
                if(start.equals("")) return false;
                if(start.charAt(start.length() - 1) == c) return true;
                return what(start.substring(0, start.length() - 1), c);
                }
}
```

7. What is the output of the following program?

```
class MyClass
{
   int i;
   float j;
   MyClass(int x, float y)
   {
      i=x;
      j=y;
   }
   public static void main(String[] args)
   {
      MyClass myObj= new MyClass();
      System.out.println("myObj.i="+myObj.i+", myObj.j="+myObj.j);
   }
}
```

A. myObj.i=0, myObj.j=0.0
B. Compile Time Error
C. Run Time Error
D. None

8. What is the output of this program?

```
public class test {
public static void main(String[] args) {
xMethod(1234567);
}
public static void xMethod(int n) {
if (n > 0) {
System.out.print(n % 10);
xMethod(n / 10);
}
}
}
```

A. 1234567 B. 7654321 C. 0 D. Error

9. What is output of the following program?

```
class Car{
        int model;
        String color;
}
public class Vehicle {
        public static void main(String[] args) {
                Car ob = new Car();
                System.out.println(ob.model + " " + ob.color);
        }
}
```

A. Compilation Error B. Run time Error C. 0 null

10. Modify the given Immutable code to Mutable code

```
// Mutable Code
public class Circle  {
 private double radius;
   public void setRadius(double radius) {
       this.radius = radius;
       }
   public double getRadius() {
       return radius;
       }
   public Circle(double radius) {
       this.radius = radius;
       }
   public double computeArea() {
       return (Math.PI * radius * radius);
       }
}
```

```
// Immutable Code
public final class Circle{
 private final double radius;
 public Circle(double radius){
 this.radius = radius;
 public double getRadius(){
     return radius;}
 public double computeArea(){
  return (Math.PI * radius * radius);
 }}
```

Part – II Programming

Problem – 1 – Class and Object

[10 Points]

A landlord owns multiple buildings, each building has multiple apartments. Each apartment has a rent associated with it. Each building generates profit which is the sum of all the apartment rents minus the building maintenance costs. Write a program that will calculate the landlord's monthly total profits.

You have to implement the two highlighted methods

a. public float getMonthlyProfit(){} in Building.java

b. public static void buildingProfit(List<Building> list){} in LanDLord.java

Here is your Partial code

```
// Apartment.java
public class Apartment {
        private String name;
        private float rentalFee;

       public Apartment(){
               this.rentalFee = 0;
               this.name = "";
       }

public Apartment(float rentalFee,String name){
               this.rentalFee = rentalFee;
               this.name = name;
```

```
// Building.java
public class Building {
    private String name;
    private List<Apartment> apartmentList;
    private float maintenanceCost;

public Building(String name,List<Apartment>
                aptList, float mtnCost){
               this.name = name;
               this.apartmentList = aptList;
               this.maintenanceCost = mtnCost;
       }
```

```
    }
    public float getRentalFee() {
        return rentalFee;
    }
    public void setRentalFee(float rentalFee) {
        this.rentalFee = rentalFee;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

```
public List<Apartment> getApartmentList() {
        return apartmentList;
    }

public float getMaintenanceCost() {
        return maintenanceCost;
    }
public float getMonthlyProfit(){
/* Implement this method.
Return the sum of all the Apartment rent minus
maintenance cost    */
}
}
```

```
// Your main class test code LanDLord.java
public class LandLord {
    public static void main(String[] args){

        List<Building> buildingList = new ArrayList<>();
        Building building1 = new Building("Building1", new ArrayList<Apartment>(){
            {
                add(new Apartment(250, "Apartment1"));
                add(new Apartment(200, "Apartment2"));
            }
        }, 100);

        Building building2 = new Building("Building2", new ArrayList<Apartment>(){
            {
                add(new Apartment(300, "Apartment1"));
                add(new Apartment(200, "Apartment2"));
            }
        }, 200);

        buildingList.add(building1);
        buildingList.add(building2);

        buildingProfit(buildingList);
    }
public static void buildingProfit(List<Building> list){
  // Print  the total Profit of Each Building, also find the sum of all building profit
}
}
```

Problem-2 – Recursion **[10 Points]**

You will find a class `SearchForString`; for this problem, you must fully implement the methods in this class. The class `SearchForString` has one instance variable `List<String> list`.

one instance variable

one constructor with signature

`SearchForString (List<String> list)`

and one instance method

`public boolean search(String s).`

The constructor should set its value into the instance variable of the class. The method `search` should be a recursive implementation of a search for the input argument `s` in the list `list`; if found, the method should return `true`; should return `false` otherwise.

Your method must implement the following recursive strategy:

- **Compare `s` to the last element of the list. If they are equal, return `true`. Otherwise, (recursively) search for `s` in the rest of the list.**

You may assume that `list` contains zero or more non-`null` `Strings`, and that the argument `s` passed in to `search` is never `null`.

To complete the problem, complete the work in the class `SearchForString` that has been provided for you here. A `private` instance method `recurSearch`, having two arguments (`s` and an integer argument `upperIndex`) has been included in `SearchForString`; it is strongly recommended that you make use of this method to do the actual recursion.

Hint: Do not forget to provide a base case for your recursion.

Note: If you solve this problem without using recursion, you will receive no credit.

```
import java.util.Arrays;
import java.util.List;

public class SearchForString {
        private List<String> list;
        public SearchForString(List<String> strings) {
                //implement
        }
        public boolean search(String s){
                //implement
                return false;
        }
        private boolean recurSearch(String s, int upper){
                //implement
                return false;
        }
        public static void main(String[] args) {
                List<String> str = Arrays.asList("Billy", "Steve", "Ralph", "Susan");
                SearchForString sfs = new SearchForString(str);
                System.out.println(sfs.search("Billy")); //expect true
                System.out.println(sfs.search("Tom")); //expect false
        }
}
```

Correct Answer for

(A) SearchForString class

```
public class SearchForString {
    private List<String> list;
    public SearchForString (List<String> strings) {
        //implimentation will be
        this.list = Strings
    }

    public boolean Search (String s) {
        // implimentation will be
        return recurSearch (s, list.size() - 1);
    }

    privte boolean recurSearch (String s, int Upper) {
        //implimentation will be.
        if (Upper == -1) {
            return false;
        } else if (list.get(upper).equals(s)) {
            return true;
        } else {
            Upper--
            return recur Search (s, Upper);
        }
    }
}
```

```
public float getMonthlyProfit(){
  float  sumOfRent = 0.0, sumOfMentCost=0.0;
for(Apartment ap : apartmentList){
   sumOfRent + = ap.getRentalFee();
   SumOfMentCost + = this.maintenanceCost;   *
}

return  sumOfRent - SumOfMentCost;
```

bus. maintenance

```
public static void buildingProfit(List<Building> List){
     float  sum = 0.0,

     for (Building bl : List){
          sum + = bl.getMonthlyProfit();
     // Prints the profit of each building.
     System.out.println( bl.getMonthlyProfit());
     }
// prints  over all profit of the buildings
     System.out.println("sum of all building
                        + sum);
}
}
```

```
public SearchForString (List<String> strings){
    List = strings;
}

Public boolean search (String s){
    return recurSearch (String s, int upper);
}

private boolean recurSearch (String s, int upper){
    if (list.size() == 0 || list.equals(null)
        retur false;
    else if (s.equals("")){
        return false;
    }
    else if (s.equals(list[upper])
        return true;
    else {
        return recurSearch (s, upper-1);
    }
}
```