CSE3211: Software Engineering
Class Test#1 (Black-Box and White Box Testing)

Q1. The New Telephone Company has the following rate structure for long distance calls:

a. Any call started at or after 6:00 p.m. (1800 hours) but before 8:00 a.m. (0800 hours) is discounted 50%.
b. Any call started at or after 8:00 a.m. (0800 hours) but before 6:00 p.m. (1800 hours) is charged full price.
c. All calls are subject to a 4% Federal tax.
d. The regular rate for a call is $0.40 per minute.
e. Any call longer than 60 minutes receives a 15% discount on its cost (after any other discount is subtracted but before tax is added).

A computer program reads the start time for a call based on a 24-hour clock and the length of the call. The gross cost (before any discounts or tax) is printed followed by the net cost (after discounts are deducted and tax is added).
The program will assume only whole number values are input, that the duration is non-negative and the start time represents a real clock time. Results are rounded to the nearest cent.

Write a complete set of Black Box test cases (equivalence classes and boundary value) for testing of the program which solves the problem above. Create a table like the one below. Include a complete description field for the purpose of the each test case.

Case #	Description	Input Values	Expected Output
1			
2			
...			

Q2. Imagine a program which reads in the length of three sides of a triangle and outputs a message naming the kind of triangle: EQUILATERAL, ISOCELES, or SCALENE. Length not in range 1 - 99 cause error message INVALID INPUT. If lengths don't make a triangle, output NOT A TRIANGLE.

Assumptions (pre-conditions for your program)

1. Three lengths are entered separated by blanks or returns.
2. Input of decimals or characters causes unpredictable results.
3. Input from keyboard, simple text output to display.
4. Even though equilateral triangle is also isosceles, only print EQUILATERAL.

Write a complete set of Black Box test cases for testing of the program which solves the problem above. Create a table like the one below.

Case #	Description	Input Values	Expected Outcomes
1	All values equal	3 3 3	EQUILATERAL
...			

Q3. What is the difference between black-box and white-box testing? During the software development, how can we derive black-box tests? How about white-box tests?

Q4. Rajshahi City Corporation is installing the AutoCop Traffic Law Enforcement System. AucoCop is a sensor-camera combo installed near a traffic light. When the sensor detects a speeding (faster than 40miles/hour) car passing by or a car running through the red light, AutoCop will activate the camera and take a picture of the plate. Use the equivalence partitioning and boundary value analysis methods to derive the test cases to test the camera activation logic.

Q5. Consider the following Java code snippet:

```
class ProductDB{
   /**
    * returns an instance of product database
    */
   public static ProductDB getInstance(){
         return null; // This line is just to remove error
   }
   /**
    * returns the price of a product.
    * throws Exception if the product is not found */
   public float getProductPrice(String productID)
               throws Exception{
         return 0.0f; // This line is just to remove error
   }
}
class Cashier{
   ProductDB db;
   public Cashier(ProductDB db){ this.db = db;
   }
   /**
    * Calculate the total of the prices of several products
    * param productIDs a String array that contains all the product
         IDs.
    * return The total price of the products.
    */
   public float calculateTotal(String[] productIDs)
               throws Exception{
         float total = 0;
         if(productIDs == null)
               return 0;
         for(int x=0; x<productIDs.length; x++){ float price =
                     db.getProductPrice(productIDs[x]);
         total += price;
         }
         return total;
   }
}
```

Q6. Consider the calculateTotal method and the following test case:

```
public void testCalculateTotal(){
   Cashier cashier = new Cashier(new MockProductDB());
   String[] products = new String[0];
   assertEquals(0, cashier.calculateTotal(products);
}
```

a) Compute the statement coverage of the test for the calculateTotal method.
b) Can we say this test achieves 100% branch coverage for the method?