To the right are abbreviations the you may use. You must write everything else in full.

Abbreviation	Stands for
S.o.p	System.out.print
S.o.pln	System.out.println
I.pI	Integer.parseInt
JOP.sID	JOptionPane.showInputDialog

Short Java API descriptions (all methods are public):

```
class Integer:
  static int parseInt(String s) // = s's value, as an int.
class Double:
  static double parseDouble(String s) // = s's value, as a double.
class Boolean:
  static boolean parseBoolean(String s) // = s's value, as a boolean.
class String:
  String substring(int i, int j) // = the letters between i (inclusive) and j (non-inclusive).
  String substring(int i) // = the letters from i (inclusive) to the end.
  int indexOf(String s) // = the index of s in this String; -1 if s is not a substring.
  int indexOf(String s, int i) // = index of s in this String after index i; -1 if s not found.
  int length() // = the number of letters in this String.
class Math:
  static double abs(double x) // absolute value of x
class JOptionPane:
  static String showInputDialog(String m) // get input from the user, prompting with m.
```

Question 1. [10 MARKS]

Below is (very stripped down) java code for class `DayPlan` and `Agenda2`. In the space provided below each `System.out.println` statement in `Agenda2`, write the output you expect to see.

```
import java.util.Date;
/**
 * DayPlan models the behaviour of a
 * single day's plan or schedule.
 */
public class DayPlan {
  private int numEvents= 0;
  private String eventList= "";
  private Date date;
  private static int numDayPlans= 0;

  public DayPlan() {
    date= new Date();
    numDayPlans= numDayPlans + 1;
  }

  public static int getNumDayPlans() {
    return numDayPlans;
  }

  public void addEvent(String event) {
    eventList= eventList + event + "\n";
    numEvents= numEvents + 1;
  }

  public int getNumEvents() {
    return numEvents;
  }
}
```

```
import java.util.Date;
/** Agenda2 test drives a DayPlan
 */
public class Agenda2 {
  public static void testDayPlan() {
    int i= 3, j= i;
    i= 4;
    System.out.println("i: " + i + " j: " + j);

    i: 4 j: 3

    int k= 5;
    System.out.println("i == k: " + (i == k));

    i == k: false

    DayPlan dp1= new DayPlan();
    DayPlan dp2= dp1;
    System.out.println("Number of day plans: " +
                       DayPlan.getNumDayPlans());

    Number of day plans: 1

    dp1= new DayPlan();
    DayPlan dp3= dp2;
    dp1.addEvent("wake up.");
    dp1.addEvent("brush teeth.");
    dp2.addEvent("sleep.");
    System.out.println("dp1 has: " + dp1.getNumEvents() +
                       " events, " + "dp2 has: " +
                       dp2.getNumEvents() + " events, " +
                       "dp3 has: " + dp3.getNumEvents() +
                       " events.");

    dp1 has: 2 events, dp2 has: 1 events, dp3 has: 1 events

    dp3= new DayPlan();
    dp2= new DayPlan();
    System.out.println("Number of day plans: " +
                       DayPlan.getNumDayPlans());

    Number of day plans: 4

    System.out.println("dp2 == dp3: " + (dp2 == dp3));

    dp2 == dp3: false
  }
}
```

Question 2. [10 MARKS]

Fill in the missing body for methods `getDay`, `getYear`, and `getMonth` of the class `License` below. Each `License` has a 17-character identifier which starts (at position 0) with an upper-case letter indicating the initial of the license-holder's surname. The last two digits of the identifier indicate the day of the month the license-holder was born. Digits at positions 10 and 12 indicate the year the license holder was born. If the license holder is male, the digits at positions 13 and 14 indicate the month of birth, otherwise these two digits indicate the month of birth **plus 50**. For example, a male license holder with date-of-birth 75/08/13 might have identifier "X1234-67897-50813", whereas a female with the same date-of-birth would have "X1234-67897-55813".

```
/**
 * a class to store and manipulate a driver's License. */
public class License {

  /**
   * License identifier, set to ridiculous default value. */
  private String identifier= "X1234-67890-23456";

  /**
   * getDay returns the day of the month that this License holder was born, as an int. */
  public int getDay() {

    String monthString= identifier.substring(15);
    return Integer.parseInt(monthString);

  }

  /**
   * getYear returns the year this License holder was born, as an int. */
  public int getYear() {

    String monthString= identifier.substring(10,11) +
      identifier.substring(12,13);
    return Integer.parseInt(monthString);
\end{verbtim}
\else %ifnosolutions
\begin{verbatim}

  }

  /**
   * getMonth returns the month this License holder was born, as an int. */
  public int getMonth() {

    String monthString= identifier.substring(13,15);
    int monthInt= Integer.parseInt(monthString);
```

```
    if (monthInt > 12) {
      return monthInt - 50;
    }
    return monthInt;

  }

}
```

Question 3. [10 MARKS]

Fill in the body of `fuzzMinus`, an instance method of class `MoreMath`. The idea of `fuzzMinus` is to subtract two doubles: `double1 - double2`. If the fractional part of `(double1 - double2` is small enough (in absolute value), return just the integer portion of `(double1 - double2)`, otherwise return (`double1 - double2`). The absolute value of `double1 - double2` (sometimes denoted |`double1 - double2`|), is the same as `(double1 - double2)` when this number is non-negative, otherwise it is −`(double1 - double2)`.

```
import javax.swing.*;
/**
 * MoreMath does a little more math.
 */
public class MoreMath {

  /**
   * fuzzMinus prompts the user for three doubles: double1, double2, and fuzz, (fuzz
   * must be non-negative).  If the fractional part of (double1 - double2) is less than
   * fuzz (in absolute value), return (double1 - double2), rounded down to the next
   * smallest integer.  Otherwise, return (double1 - double2).
   *  You may find Math.abs() useful.  */
  public double fuzzMinus() {
    double double1=
      Double.parseDouble(JOptionPane.showInputDialog("Enter first double:"));
    double double2=
      Double.parseDouble(JOptionPane.showInputDialog("Enter second double"));
    double fuzz=
      Double.parseDouble(JOptionPane.showInputDialog("Enter third double:"));
    if(Math.abs((double1 - double2) - (int) (double1 - double2)) < fuzz) {
      return (int) (double1 - double2);
    }
    else {
      return double1 - double2;
    }

  }

  // This solution is also correct, since the method comment and the question
  // description aren't entirely consistent (round down to nearest integer
  // versus keep the integer portion).
  public double fuzzMinus() {
    double double1=
      Double.parseDouble(JOptionPane.showInputDialog("Enter first double:"));
    double double2=
      Double.parseDouble(JOptionPane.showInputDialog("Enter second double"));
    double fuzz=
      Double.parseDouble(JOptionPane.showInputDialog("Enter third double:"));
    if (Math.abs((double1 - double2) - (int)(double1 - double2)) < fuzz) {
      if ((double1 - double2 >= 0) ||
```

```
          ((double1 - double2) - (int)(double1 - double2) == 0)) {
        return (int) (double1 - double2);
      }
      else {
        return (int) (double1 - double2) - 1;
      }
    }
    else {
      return double1 - double2;
    }
  }

}
```

Total Marks = 30