CS100J, Spring 2005, Assignment A3.
Due 03 March before midnight, submitted to the CMS

PURPOSE.
This assignment gives you practice with writing functions whose bodies have assignment statements, conditional statements, and blocks.

GROUND RULES.
You may work with one partner. If you are going to work with a partner, form your group on the CMS BEFORE you submit the assignment. Form the group at least 3 days before you submit so that problems can be fixed easily.

This is (roughly) a 10-hour assignment. Plan accordingly. Start early. Let's see how many people can submit the project 3 days before the deadline. Sspend as much as an hour reading this handout so that you thoroughly understand what we are asking for. Do this **with your partner** before you start programming! You and your partner should alternate writing the methods. You will both benefit from this.

Begin by downloading class `DatesMethods` and putting it in a new directory for this assignment. Study its methods `month` and `numberOfdays`.

Submit `DatesMethods.java` and `DatesMethodsTester.java` on the CMS by the due date.

VITAL RULES
You must use the capitalization that we specify. You can ensure this by copying identifiers from this handout, rather than typing them yourself. Also, write precise specifications of your functions. Copy-and-paste makes this easy.

Besides class `DatesMethods`, you will be submitting a JUnit class `DatesMethodsTester` that tests your methods. Create this class using menu `File` item `New JUnit Test Case..`. Here is how we expect you to **write and test each method**, in turn.

1. Write a stub for the method --its specification, its heading, and a body with one statement --a return statement that returns 0 or null), just so the program compiles.
2. Based on the specification, write down a set of test cases that you think you will need in order to test the program thoroughly.
3. Write a test method in class `DatesMethodsTester` that tests the test cases given in point 2.
4. NOW write the body of the method.
5. Test and debug until all your test cases run properly.

CLASS DATESMETHODS
This assignment is presented as a series of tasks, each being to write a function in class `DatesMethods`. Be sure that each one has a javadoc comment --your grade depends partially on the javadoc specs. Here are the tasks:

Task 1. Make a big leap. Method `month` does not take leap years into account. Write and test a new function `numberOfDays` with the specification shown below. Do not change the old function! Define a new one. Thus, the function name will be "overloaded". Write the specification of the function before writing the function! You can copy our specification.

A leap year is a year that satisfies two properties: (1) it is divisible by 4 and (2) if it is divisible by 100, it is also divisible by 400. Thus, 2000 is a leap year, but 1900, 2100, 2200, and 2300 are not leap years. This web page explains the reason for the second property. In a leap year, February has 29 days instead of 28.

```
/** = the number of days in month m of year y. Precondition: 1 <= m <= 12) */
public static int numberOfDays(int m, int y)
```

Task 2. Which month is longer? Develop and test the following method. Remember to implement test cases in a separate method of class `DatesMethodsTester`. This method should not have to worry about leap years or anything like that, because it will rely on your method `numberOfDays(m,y)`.

```
/** = "month m1 in year y1 has more days than month m2 in year y2."
       Precondition 1 <= m1 <= 12, 1 <= m2 <= 12 */
public static boolean longerMonth(int m1, int y1, int m2, int y2)
```

Task 3. Are the months the same length? Develop and test the following method. Remember to implement test cases in a separate method of class `DatesMethodsTester`. This method should not have to worry about leap years or anything like that, because it will rely on your method `numberOfDays(m,y)`.

```
/** = "month m1 in year y1 has the same number of days as month m2 in year
y2".
```

```
       Precondition 1 <= m1 <= 12, 1 <= m2 <= 12 */
public static boolean sameLength(int m1, int y1, int m2, int y2)
```

Task 4. Make a date: Write a function with the following specification.

```
/** = the date for day d of month m of year y .
      For example, date(2, 3, 200) = "2 March 2000". Note that precisely one
      blank separates the day from the month and the month from the year.
      Precondition: day d, month m, year y is a valid date */
public static String date(int d, int m, int y)
```

Task 5. Determine the month. Write function `theMonth`, which is specified below. Be sure to use calls on earlier written functions where appropriate. A good programmer uses already written and checked out methods to simplify later tasks. For example, the body of this function should **NOT** contain the constant 31 for the number of days in January, and it should **NOT** worry about leap years.

```
/** = the month in which day d of year y occurs.
      E.g. day 32 occurs in month 2 (February) and
      day 360 occurs in month 12 (December).
      Precondition: 1 <= d <= (no. of days in year y) */
public static int theMonth(int d, int y)
```

The body of `theMonth` will probably consist of repetitive code --something like this:

> if d occurs in January then return 1;
> if d occurs in February then return 2;
> if d occurs in March then return 3;
> etc.

In order to make this easy to do, we suggest that you use two local variables `m1` and `daysBefore` with the following meaning:

(1) `m1` is a month, and day `d` does not occur before month `m1`
(2) `daysBefore` is the number of days in the year that precede month `m1`,

Truthify this relation initially with the assignments `m1= 1; daysBefore= 0;`. If you find out that `d` does not occur in January, then add `1` to `m1` and change `daysBefore` accordingly.

This is a tough function to write. Make sure you provide enough test cases to give you assurance that it is correct. We also suggest getting the code correct for January and February first. The rest of them can be done by copying and pasting and then editing a bit.

Task 6. Determine the day of the year. Write function `dayOfYear`, specified below. For example, `dayOfYear(2,2,2000) = 33`.

```
/** = the day number within the year of day d, month m, year y.
      Precondition: day d, month m, year y is a valid date */
public static int dayOfYear(int d, int m, int y)
```

The body of this function will probably consist of repetitive code --11 or 12 copies of the same thing, with minor changes, as in the previous function. So, copy-paste-change is a good way develop the body. Look at the hint for the previous function.

Task 7. Determine the day of the month. Write a function with the following specification. Do **not** write repetetive code, as in the previous task. Instead, rely on previously written functions.

```
/** = day of the month in which day d of year y occurs.
      E.g. theDay(32,2000) = 1, since day 32 is 1 Feb., and theDay(366,2000) =
31.
      Precondition: 1 <= d <= (number of days in year y) */
public static int theDay(int d, int y)
```

Task 8. Change the format. Write a function with the following specification:

```
/** Precondition: s contains a date, like this one: 15 February 2005
There is exactly 1 space between the day and month and between the
month and year.
Return the same date but in this form: February 15, 2005
There should be exactly 1 blank between the month and day and
between the "," and year. */
public static String format(String s)
```