UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

**CS61B** **M. Brudno**
**Summer 2000**

# Homework #2

**Due:** Wednesday, July 5, at 11:00am

Create a directory to hold your answers to this homework set. You will find templates for some problems in `~cs61b/hw/hw2`; please use them. To submit your homework, use the command '`submit hw2`' from within that directory. Put the answer to 2c and 3 in a file called `hw2.txt`.

**1.** You will implement a simple Date class. It will support the following operations:

```
/** Constructs a Date with the given month, day and year. If the date is
 *  not valid, the entire program will halt with an error message.
 */
public Date(int month, int day, int year) {
}

/** Constructs a Date object corresponding to the given string.
 *  s should be a string of the form "month/day/year" where month must
 *  be one or two digits, day must be one or two digits, and year must be
 *  between 1 and 4 digits.  If s does not match these requirements or is not
 *  a valid date, the program halts with an error message.
 */
public Date (String s) {
}

/** Checks whether the given year is a leap year.
 */
public static boolean isLeapYear(int year) {
}

/** Returns the number of days in a given month.
 */
public static int daysInMonth(int month, int year) {
}

/** Checks whether the given date is valid.
 */
public static boolean isValidDate(int month, int day, int year) {
  return true;             // replace this line with your solution
}

/** Returns a string representation of a date in the form month/day/year.
 *  The month, day, and year are printed in full as integers; for example,
 *  12/7/1998 or 3/21/407.
 */
public String toString() {
  }

  /** Determines whether this Date is before the Date d.
   *  Returns true if and only if this Date is before d.
   */
  public boolean isBefore(Date d) {
  }

  /** Determines whether this Date is after the Date d.
   *  Returns true if and only if this Date is after d.
   */
  public boolean isAfter(Date d) {
  }

  /**  Returns a number n in the range 1...366, inclusive, such that this Date
   *  is the nth day of its year.  (366 is only used for December 31 in a leap
   *  year.)
   */
  public int dayInYear() {
  }

  /** Determines the number of days between d and this Date.  For example, if
   *  this Date is 12/15/1997 and d is 12/14/1997, the difference is 1.
   *  If this Date occurs before d, the result is negative.
   */
  public int difference(Date d) {
  }
```

**2.** The following problems are based on the famous (or infamous) Towers of Hanoi. It is said that somewhere in the far east there is a Buddhist monastery. Inside the monastery are three poles, on the poles there are 64 golden rings. The rings are all of a different radius, and a ring with a larger radius can never be placed on top of the ring with a smaller radius. The rings were at first stacked in order (largest on the bottom, smallest on the top) on one of the poles. Ever since then, diligent monks have been moving rings from pole to pole, trying to get all of the rings onto another pole. Once they finish their task the world will end. Your job will be to mimick their selfless task. Your program will start with all of the rings on pole 1, and will finish once all of them have moved to pole 3. The program will take a single commandline argument, the number of rings. If the program was run as `java Towers 2` it should print

```
1->2
1->3
2->3
```

The movements should be accomplished in the minimum number of steps.

a. Make a class `Towers` with a main program that will take a single integer argument, the number of rings, and print the appropriate sequence of ring movements. Your underlying algorithm should be recursive.

b. Repeat part a., but place make your solution iterative and place it in the class `Towers2`. This problem is reasonably tricky. You will have to think quite a bit about the algorithm, and may find the `java.util.Stack` class useful.

c. If the monks started their task on the 1st of January in the year 0, and assuming they move a ring every second, when will the world end?

**3.** We are going to design a Date Planner. The date planner will be able to keep a list of appointments, birthdays, alarms, things to do, and the like. Please decide whether each of the following should be a regular class, an abstract class, an interface, or whether it should not have an object representation. Give a 1 or 2 line reason for each.

- Event – A generic Event
- TimedEvent – An Event which will happen at some specific time
- Timer – Tells you whether a certain time has come
- Birthday – Someone's birthday
- Todo – something todo. May include a deadline
- EventChecker – checks whether there are any events coming up
- Day – List of events for a specific day