CS 255 – Spring 2013
Homework 2

Due: **Thursday, Feb. 7th by the start of class.**

This is an individual assignment. See the course website for appropriate collaboration policies. Include your collaboration statement as a comment at the top of your code file.

"Novendecimal" Numbers

The "Novendecimal" numbers are based on the number 19. The digits used in this numbering system are:

- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, !, @, #, $, %, ^, &, *, and ?

The values of the digits are 0 to 18, where ! = 10, @ = 11, # = 12, $ = 13. % = 14, ^ = 15, & = 16, * = 17, and ? = 18.

Also, when a number contains multiple digits, then the value of a digit 19 times its predecessor digit.

Examples:

```
Novendecimal Number            Value represented
-------------------            ------------------
       1!                        29 (1 x 19 + 10)
       2?                        56 (2 x 19 + 18)
```

Assignment

Write a Java class "NovenNumber" that contains the `static` methods `parseNoven(String s)` and `toString(int a)` which allows you to convert between a "novendecimal" number **string** to integer numbers.

The class definition looks like this:

```
public class NovenNumber {
   public static int parseNoven(String s) {
      ...
    }

   public static String toString(int a) {
      ...
    }
 }
```

The `NovenNumber.parseNoven(String s)` method converts a "novendecimal" number string into an integer number.

For example:

- `NovenNumber.parseNoven("!")` returns the integer number 0000000000001010 in binary (which represents the **value** 10)
- `NovenNumber.parseNoven("?")` returns the integer number 0000000000010010

in binary (which represents the **value** 18)

- `NovenNumber.parseNoven("10")` returns the integer number 0000000000010011 in binary (which represents the **value** 19)

The `NovenNumber.toString(int a)` method converts an integer number to a "novendecimal" number string.

For example:

- `NovenNumber.toString(10)` returns the string "!"
- `NovenNumber.toString(16)` returns the string "&"
- `NovenNumber.toString(19)` returns the string "10"

Restrictions and Clarifications:

- You cannot use any of the Integer.parseInt() methods to write your code. The point of this assignment is to reinforce understanding of conversion between strings and binary representation. The assignment is not to test "how well" you can use Java's library! If you use Java library calls in the 2 methods, your assignment will receive a very low score.

Correcting misconceptions:

- Many students think that because we write `NovenNumber.toString(10)`, `NovenNumber.toString(16)` and `NovenNumber.toString(19)` that the `NovenNumber.toString()` method uses a decimal number as a parameter.

  This is not the case.

  The type `int` in Java contains a binary number using 2's complement encoding to represent signed values.
- Also, every number that you see inside a Java (or any programming language) is converted into binary by the compiler.
- So: in `NovenNumber.toString(10)`, the decimal number 10 is converted into a binary number (000...0001010).

  There is not a single decimal value left inside a program in machine code. This is also true in an Assembler Program

Test Program

- Get a copy of the test program **hw2_check.java** from the **~cs255000/share/hw2** directory and put it in your cs255 project directory.
- The test program reads in 2 base-19 numbers (as Strings) from the terminal and invokes parseNoven() to convert them into binary to perform addition. This test program will print the (binary value) returned by parseNoven() in decimal form so you can check the correctness of parseNoven(). The sum is then converted back from binary to String by toString() for printing.
- A simpler test program is **hw2.java** - this version will **not print** the value returned by the **parseNoven()**. You can get this program from the ~cs255000/share/hw2 directory.
- Neither of these programs is a standalone program! You must first write your NovenNumber.java class with the 2 appropriate static methods in it. Then you can compile and

run either hw2_check.java or hw2.java.

4. Turn in

Turn in your `NovenNumber.java` program as hw2 using the `/home/cs255000/turnin-hw` command (execute the turnin-hw command while you're in your cs255 directory):

```
/home/cs255000/turnin-hw  NovenNumber.java  hw2
```

You should only submit the source (ie the .java) file. **DO NOT** turn in the object (ie the .class) file !