

## Base Conversion

One algorithm for converting a base **10** number to base **b** involves repeated division by the base **b**. Start by dividing the number by **b**. The remainder from this division is the units digit (the rightmost digit) in the base **b** representation of the number. The quotient is then divided by **b** on the next iteration. The remainder from this division gives the next base **b** digit from the right. The quotient from this division is used in the next iteration. The algorithm stops when the quotient is **0**. Note that at each iteration the remainder from the division is the next base **b** digit from the right -- that is, this algorithm finds the digits for the base **b** number in reverse order.

Here is an example for converting **30** to base **4**:

	<u>quotient</u>	<u>remainder</u>
<b>30 / 4 =</b>	<b>7</b>	<b>2</b>
<b>7 / 4 =</b>	<b>1</b>	<b>3</b>
<b>1 / 4 =</b>	<b>0</b>	<b>1</b>

The answer is read bottom to top in the remainder column, so

$$\mathbf{30 \text{ (base 10) = 132 \text{ (base 4)}}$$

Think about how this could be done with **recursion**: If you want to convert **x** ( **30** in the example ) to base **b** ( **4** in the example ), the rightmost digit is the remainder **x % b**. To get the rest of the digits, you perform the same process on what is left; that is, you convert the quotient **x / b** to base **b**. If **x / b** is **0**, there is no rest; **x** is a single base **b** digit and that digit is **x % b** (which also is just **x**).

The file **BaseConversion.java** contains the shell of a method `convert` to do the base conversion and a main method to test the conversion. The `convert` method returns a string representing the base **b** number, hence for example in the base case when the remainder is what is to be returned it must be converted to a **String** object. This is done by concatenating the remainder with a null string. The outline of the `convert` method is as follows:

```

public static String convert (int num, int b)
{
    int quotient; // the quotient when num is divided by base b
    int remainder; // the remainder when num is divided by base b

    quotient = _____;

    remainder = _____;

    if ( _____ ) //fill in base case
    {
        return (" " + _____);
    }
    else
    {
        // Recursive step: the number is the base b representation of
        // the quotient concatenated with the remainder

        return (
_____);
    }
}

```

Fill in the blanks above, then in **BaseConversion.java** complete the method and main. Main currently asks the user for the number and the base and reads these in. Add a statement to print the string returned by **convert** (appropriately labeled).

Test your function on the following input:

- \* **Number: 89**      **Base: 2** ---> **should display 1011001**
- \* **Number: 347**    **Base: 5** ---> **should display 2342**
- \* **Number: 3289**   **Base: 8** ---> **should display 6331**