Recursion in Java – Introduction to Recusion

Everyday Recursion

To draw a family tree, we start with an individual and draw all direct descendants. You might start by writing the name of the person, and under that name write the names of any children. For each child, you would then write the names of their children, and so on, until you reach the various dead-ends of the tree (i.e., individuals without children).

This process can be described easily and efficiently using recursion.

```
To draw the family tree of a person
      write the name of the person
      if the person had children
            for each child
                  draw the family tree of the child
```

This example illustrates two features typical of recursive algorithms.

1. The simplest case of the problem can be solved directly, without recursion. In our example, a childless person is the simplest case; after writing their name, that person's family tree is complete.

2. All other cases are done by writing the name of the current person, and then addressing their children using the same algorithm.

There are many examples of this recursive approach to solving problem in real life, although we are often unaware of them at a conscious level.

In any recursive algorithm, it is crucial that we have a way of stopping the process. This is often called a *terminating condition* of a recursive algorithm. In our previous example, the terminating condition is when a person has no children.

Exercises

1. The following is a recursive definition of the process of climbing the stairs.

   ```
   To climb the stairs
         if you are at the top
               you are done
         else
               take one step up
               climb the stairs
   ```

   (a) Identify the simple case in climb the stairs
   (b) Identify the complex case in climb the stairs

2. Why is the terminating condition so important in a recursive algorithm?

3. Which of the following can be said to be examples of recursion?
   (a) closing a zipper
   (b) nested Russian wooden dolls
   (c) a reflection of a mirror in a mirror

4. Rewrite the definition of climb the stairs using a while statement.

5. Give a recursive description of the process of reading a book.

6. Find another everyday task that can be defined recursively and give the recursive algorithm for it.

Recursion in Mathematics

Suppose we have the following sequence of integers:

3, 6, 12, 24, ...

in which any term is twice as large as the preceding one. If we call the terms t_1, t_2, t_3, ... then we can express the relationship concisely as follows:

$$t_1 = 3$$
$$t_n = 2 \times t_{n-1}, \quad if\ n > 1$$

This definition of the terms of the sequence satisfies the conditions that are required for any recursive definition:

1. An object is defined in terms of another object of the same type. Here a term is defined in terms of the preceding term.

2. There is some way of stopping the recursion. Here, this is done by explicitly defining the first term, t_1, as having the value 3.

To find the fifth term in the sequence, t_5, we refer to the previous term, t_4., which is 24. Thus t_5 is 48. Using this recursive definition, we can theoretically find the value of any term.

One of the most famous recursive algorithms was stated over two thousand years ago by the Greek mathematician Euclid. *Euclid's algorithm*, as it is called, provides a method for finding the *greatest common divisor* (gcd) for a pair of natural numbers. The algorithm is based on the following properties of a gcd:

1. The gcd of two equal values, *m* and *n*, is simply *m* (or *n*, since they are the same).

   e.g., The gcd of 7 and 7 is simply 7, which is the largest number that will divide into 7.

2. If two numbers are not equal, then the gcd of the two values will also divide into their difference and will, in fact, be the gcd of this difference. Thus, if *m* > *n*, then the gcd of *m* and *n* will be equal to the gcd of *n* and *m* - *n*.

   e.g., The number 15 and 10 have a difference of 5. Therefore, the gcd of 15 and 10 must also be the gcd of 10 and 5 (which is, of course, 5).

3. If the numbers are not equal and the first number is smaller than the second, then we can simply switch the numbers and apply rule 2.

These rules can be stated more concisely, and more clearly, in a symbolic form:

1. If m = n, then gcd(m, n) = m

2. If m > n, then gcd(m, n) = gcd(n, m - n)

3. If m < n, then gcd(m, n) = gcd(n, m)

To see this algorithm in action, consider the values of 54 and 90.

gcd(54, 90)	=	gcd(90, 54)	swap *m, n* by rule 3
	=	gcd(54, 36)	find the gcd using the difference, by rule 2
	=	gcd(36, 18)	find the gcd using the difference, by rule 2
	=	gcd(18, 18)	find the gcd using the difference, by rule 2
	=	18	by rule 1

Exercises

1. Write the first five terms of each sequence (correct to two decimal places).

   (a)
   $$t_1 = 3$$
   $$t_n = t_{n-1} + 2, n > 1$$

   (b)
   $$t_1 = 2$$
   $$t_{n+1} = 1 - \frac{1}{t_n}, n > 0$$

   (c)
   $$t_1 = 2$$
   $$t_{n+1} = n + t_n, n > 0$$

   (d)
   $$t_1 = 1$$
   $$t_2 = 2$$
   $$t_n = t_{n-1} + t_{n-2}, n > 2$$

2. Write a possible recursive definition for each sequence.

   (a) 10, 13, 16, 19, ...
   (b) 5, 15, 45, 135, ...
   (c) 256, 64, 16, 4, ...
   (d) $\frac{1}{2}$, $\frac{3}{2}$, $\frac{5}{2}$, ...

3. Use Euclid's algorithm to find the gcd of each pair of numbers.

   (a) 18 and 24
   (b) 84 and 144
   (c) 35 and 16
   (d) 515 and 206

4. Euclid's algorithm

   (a) If we were to use Euclid's algorithm to find the gcd of 44 and 6, how many subtractions would be required to reduce the problem to that of finding the gcd of 6 and 2?

   (b) What operation would achieve the same effect as this sequence of subtractions?

   (c) Suggest a modification of the algorithm that avoids this sequence of subtractions.

   (d) Use the modified algorithm to find the gcd of each pair of numbers in question 3.