Chapter 11: Recursion

- Recursion is a fundamental programming technique that can provide an elegant solution to certain kinds of problems.
- Chapter 11 focuses on:
  - Thinking in a recursive manner
  - Programming in a recursive manner
  - The correct use of recursion
  - Recursion examples

Recursive Thinking

- A recursive definition is one which uses the word or concept being defined in the definition itself.
- When defining an English word, a recursive definition is often not helpful.
- But in other situations, a recursive definition can be an appropriate way to express a concept.
- Before applying recursion to programming, it is best to practice thinking recursively.

Recursive Definitions

Consider the following list of numbers:
24, 88, 40, 37

- Such a list can be defined as:
  A LIST is a: number
  or a: number comma LIST

- That is, a LIST is defined to be a single number, or a number followed by a comma followed by a LIST.
- The concept of a LIST is used to define itself.

Infinite Recursion

- All recursive definitions have to have a non-recursive part.
- If they didn’t, there would be no way to terminate the recursive path.
- Such a definition would cause infinite recursion.
- This problem is similar to an infinite loop, but the non-terminating “loop” is part of the definition itself.
- The non-recursive part is often called the base case.

Recursive Definitions

N!, for any positive integer N, is defined to be the product of all integers between 1 and N inclusive.

- The recursive part of the LIST definition is used several times, terminating with the non-recursive part:

  number comma LIST
  24 , 88, 40, 37
  number comma LIST
  88 , 40, 37
  number comma LIST
  40 , 37
  number

Recursive Definitions

- N!, for any positive integer N, is defined to be the product of all integers between 1 and N inclusive.
- This definition can be expressed recursively as:

  \[ 1! = 1 \]
  \[ N! = N \times (N-1)! \]

- The concept of the factorial is defined in terms of another factorial.
- Eventually, the base case of 1! is reached.
Recursive Definitions

5!
5 * 4!
4 * 3!
3 * 2!
2 * 1!
1

Recursive Programming

• A method in Java can invoke itself; if set up that way, it is called a recursive method.
• The code of a recursive method must be structured to handle both the base case and the recursive case.
• Each call to the method sets up a new execution environment, with new parameters and local variables.
• As always, when the method completes, control returns to the method that invoked it (which may be an earlier invocation of itself).

Consider the problem of computing the sum of all the numbers between 1 and any positive integer N.

This problem can be recursively defined as:

\[ \sum_{i=1}^{N} i = N + \sum_{i=1}^{N-1} i = N + (N-1) + \sum_{i=1}^{N-2} i = \text{etc.} \]

See Recursive_Sum.java

Recursive Programming

• Consider the problem of computing the sum of all the numbers between 1 and any positive integer N.
• This problem can be recursively defined as:

Recursive Programming

• A method invoking itself is considered to be direct recursion.
• A method could invoke another method, which invokes another, etc., until eventually the original method is invoked again.
• For example, method m1 could invoke m2, which invokes m3, which in turn invokes m1 again.
• This is called indirect recursion, and requires all the same care as direct recursion.
• It is often more difficult to trace and debug.
Indirect Recursion

Using Recursion
- A maze is solved by trial and error -- choosing a direction, following a path, returning to a previous point if the wrong move is made
- As such, it is another good candidate for a recursive solution
- The base case is an invalid move or one which reaches the final destination
- See MazeSearch.java
- See SolveTowers.java

Using Recursion
- Recursion is best served when it is easy to define a smaller subset of the problem in terms of the original
- Consider the task of repeatedly displaying a set of images in a mosaic that is reminiscent of looking in two mirrors reflecting each other
- The base case is reached when the area for the images shrinks to a certain size
- See MirroredPictures.java