

Recursion -- Introduction

- Recursion is a fundamental programming technique that can provide an elegant solution certain kinds of problems
- Chapter 12 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 concept being defined in the definition
- When defining an English word, a recursive often not helpful
- But in other situations, a recursive definition appropriate way to express a concept
- Before applying recursion to programming, 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 followed by a comma followed by a LIST
- The concept of a LIST is used to define its

Recursive Definitions

- The recursive part of the LIST definition is terminated 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
```

```
37
```

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
- Terminating "loop" is part of the definition
- The non-recursive part is often called the base case

Recursive Definitions

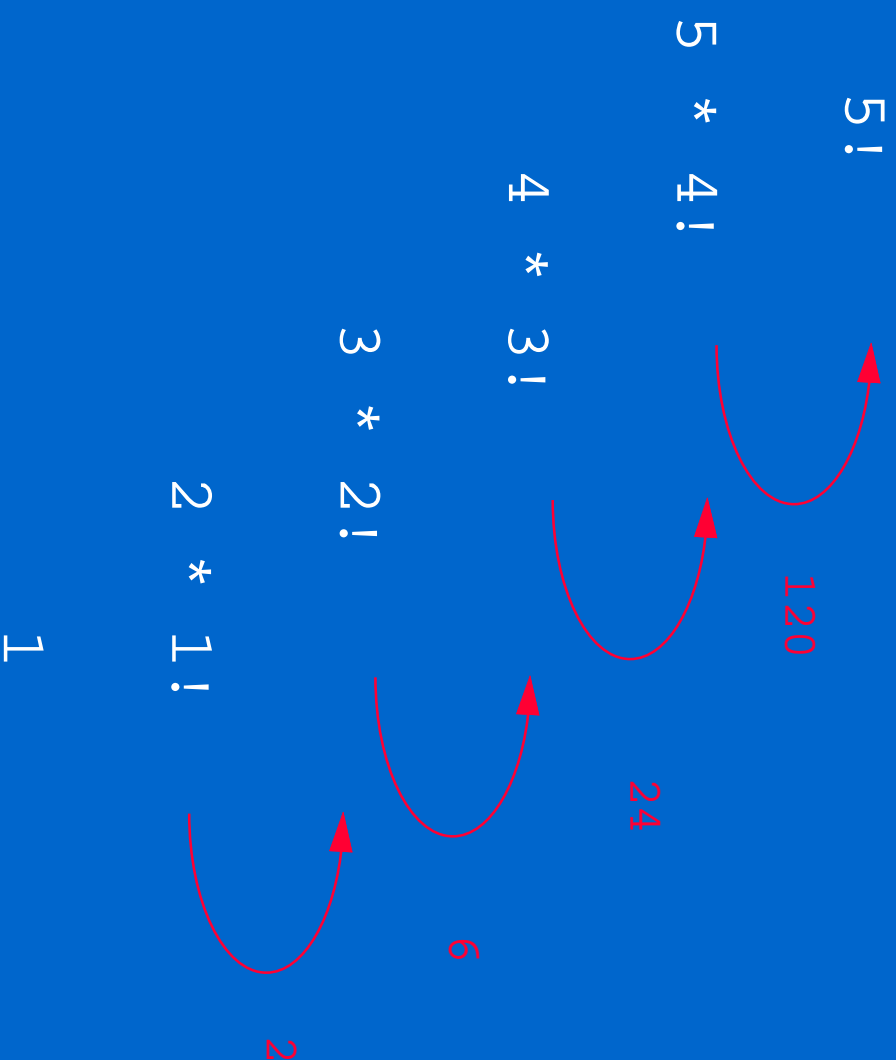
- $N!$, for any positive integer N , is defined product of all integers between 1 and N inclusive
- This definition can be expressed recursively

$$1! = 1$$

$$N! = N * (N-1)!$$

- The concept of the factorial is defined in another factorial
- Eventually, the base case of $1!$ is reached

Recursive Definitions



Recursive Programming

- A method in Java can invoke itself; if set 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 the method itself)

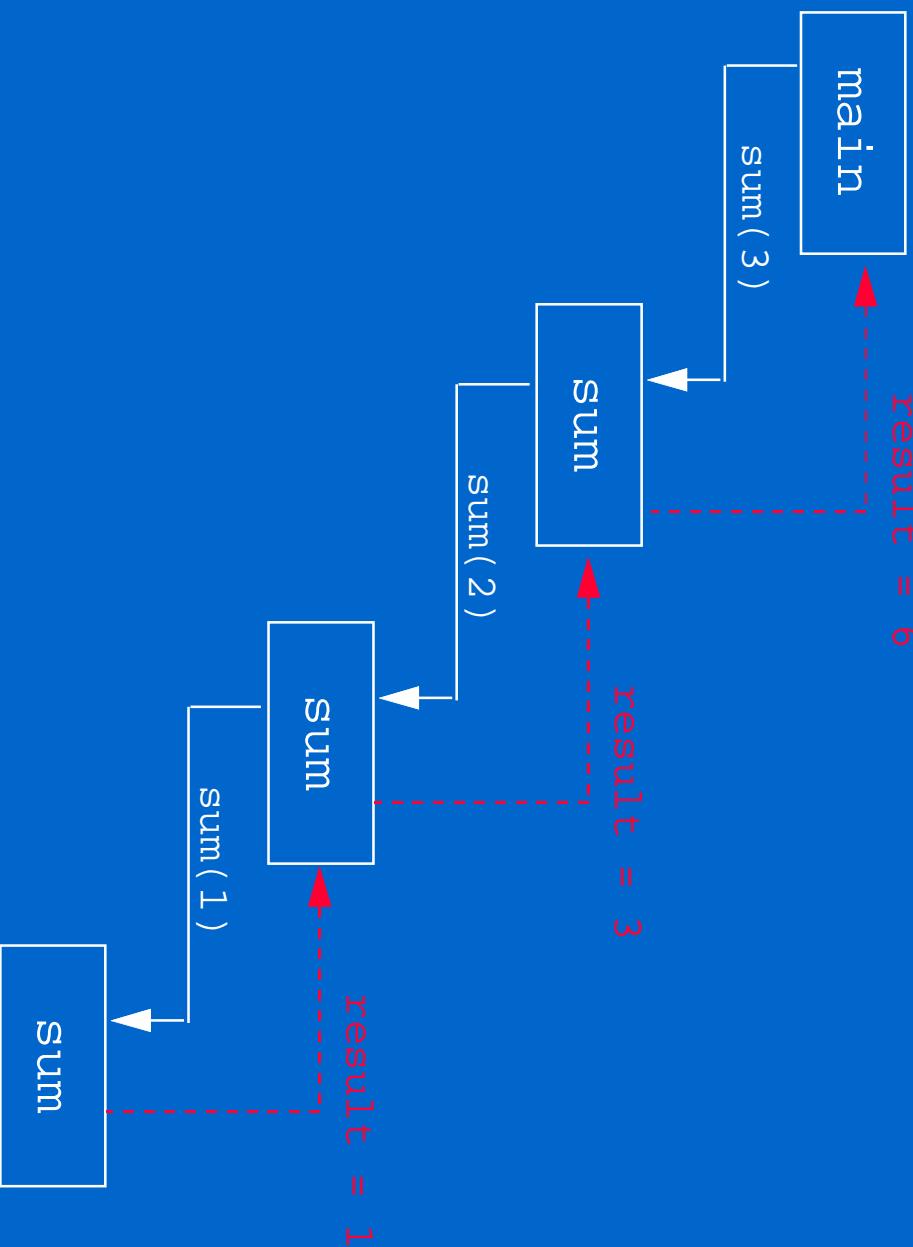
Recursive Programming

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

$$\begin{aligned} \sum_{i=1}^N &= N + \sum_{i=1}^{N-1} = N + (N-1) + \sum_{i=1}^{N-2} \\ &= \text{etc.} \end{aligned}$$

- See `Recursive_Sum.java`

Recursive Programming



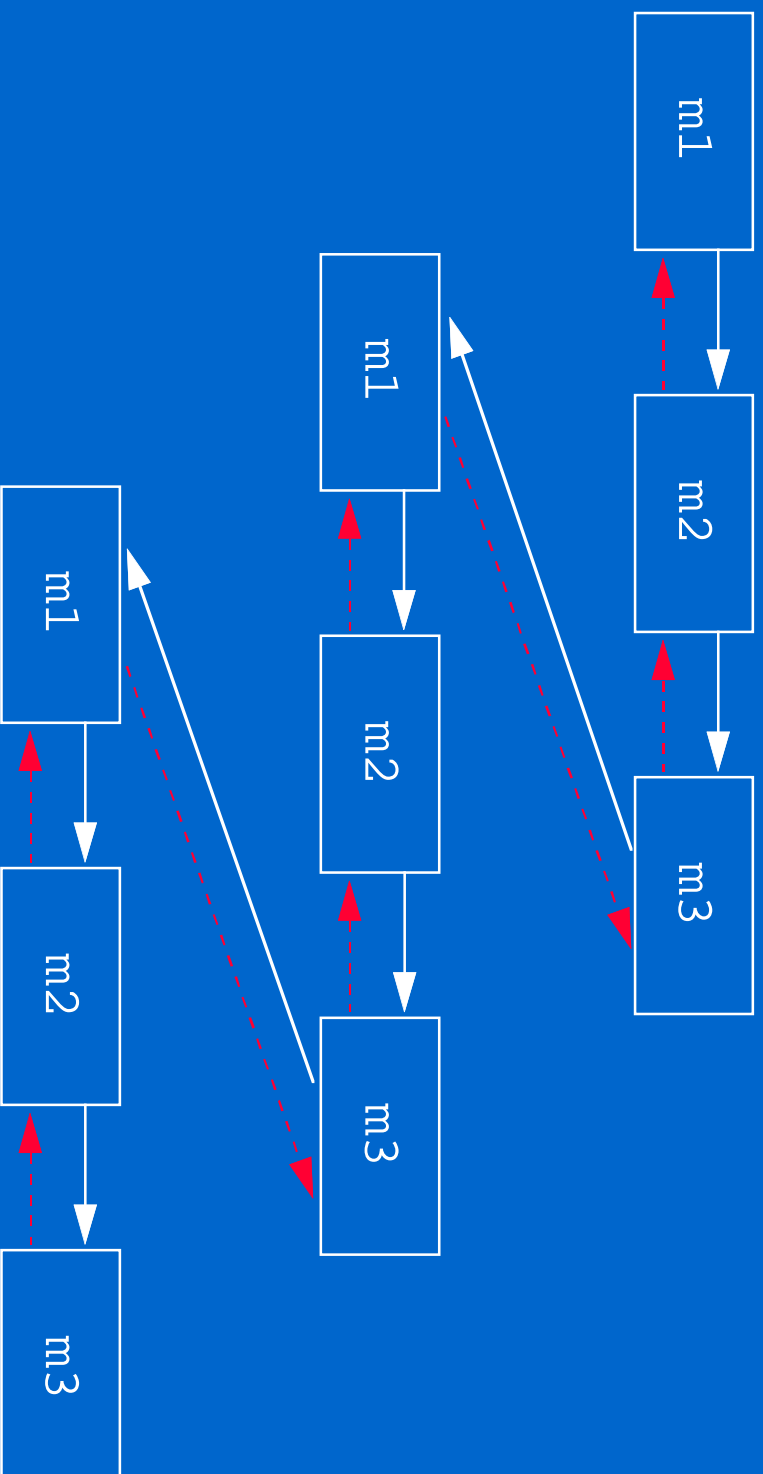
Recursive Programming

- Note that just because we can use recursion for a problem, doesn't mean we should
- For instance, we usually would not use recursion for the sum of 1 to N problem, because the iterative solution is easier to understand
- However, for some problems, recursion provides a more elegant solution, often cleaner than an iterative solution
- You must carefully decide whether recursion is the correct technique for any problem

Indirect Recursion

- A method invoking itself is considered to be 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 the same care as direct recursion
- It is often more difficult to trace and debug

Indirect Recursion



Using Recursion

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

Using Recursion

- A palindromes a string of characters that read the same forward and backward:
 - radar
 - able was I ere I saw elba
- To determine whether a string is a palindrome, examine the two outer characters, and work toward the middle of the string
- This solution is easily defined recursively
- See Palindromes.java

Using Recursion

- A maze is solved by trial and error -- choose a direction, following a path, returning to a previous direction 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 `Maze_Search.java`