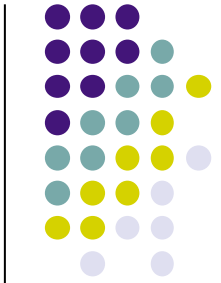
SumDigits6

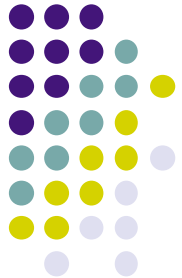


- Sum of digits of a six-digit positive integer

```
fun {SumDigits6 N}
   {SumDigits (N div 1000)} +
   {SumDigits (N mod 1000)}
end
```

- This is an example of function composition: defining a function in terms of other functions
  - This is a key ability for building large systems: we can build them in layers, where each layer is built by a different person
  - This is the first step toward data abstraction

SumDigitsR

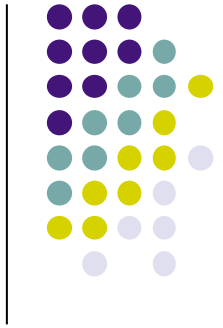


- Sum of digits of any positive integer (first try)

```
fun {SumDigitsR N}
   (N mod 10) + {SumDigitsR (N div 10)}
end
```

- This function calls itself with a smaller value
  - But it never stops: we need to make it stop!

SumDigitsR

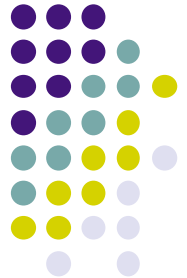


- Sum of digits of any positive integer (correct)

```
fun {SumDigitsR N}
   if (N==0) then 0
   else
      (N mod 10) + {SumDigitsR (N div 10)}
   end
end
```

- This introduces the conditional (**if**) statement
- This is an example of function recursion: defining a function that calls itself
  - This is a key ability for building complex algorithms: we divide a complex problem into simpler subproblems (divide and conquer)

The functional paradigm



- The functional paradigm as we have introduced it now, with the ability to calculate with numbers, to define functions, to do function composition and recursion, and to use the conditional statement (**if**), is a fully capable programming language

- We say it is Turing complete, since it can compute the same functions as a Turing machine
  - Since a Turing machine is the most powerful computer we know how to build (in terms of the kinds of functions that can be programmed), this means that we can do anything that any other computer can do

- We will see how to harness the power of recursion in the next two lessons (invariant programming and symbolic programming)
  - We will continue to harness the power of functions in the rest of the course (higher-order programming, data abstraction, concurrent programming)