Explain the One-at-a-Time Rule.

How can you circumvent it?

Only one implicit is tried.

`x + y` will never be rewritten.

```
convert1(convert2(x)) + y
```

Circumvent this with implicits take implicit parameters.

Explain the Explicits-First Rule.

Give a corollary.

Whenever code type checks as it is written, no implicits are attempted.

Corollary: you can always replace implicit identifiers with explicit ones to make code longer but clearer.

Explain the Marking Rule.

Only definitions marked implicit are available.

Explain the Non-Ambiguity Rule.

An implicit conversion is only inserted if there is no other possible conversion to insert.

There is no "best match" rule.

Give the most common reason for using view bounds.

Use when implicit parameter is never explicitly used in method body, but only present to introduce implicit conversion ability to body.

```
def maxList[T] (elements: List[T])
  (implicit orderer: T => Ordered[T]):T
def maxList[T <% Ordered[T]]
  (elements: List[T]):T
```

Why do implicit conversion methods have names?

- Explicit invocation.
- Selectively importing one conversion method but not another.

What does Scope Rule help with?

Modular reasoning. You only need to consider code that is imported or explicitly referenced through a fully qualified name.

Give the most common use of implicit parameters.

They provide information about a type mentioned explicitly in the earlier parameter list.

What is the fundamental difference between implicit and 2.8's default parameters?

What's the consequence?

Default arguments are controlled at the point of definition.

Implicit parameters are chosen at the point of use and, as such, cannot be called in short form without the implicit being defined.

Give some tips for debugging implicits.

- Try using conversions explicitly to get a new error message or to establish scoping problem.

- Use `-Xprint:typer` option to see your code after its implicits have been expanded.

Where are implicits tried?

- Conversion to an expected type (e.g. to match a parameter).
- Conversions of the receiver of a selection (e.g., "abc".exists(...)`).
- Implicit parameters.

Give the convention for libraries providing implicit conversions.

```
Preamble object.
import Preamble._
```

Where can implicit keyword be used?

Any variable, function, or object definition.

Describe the Scope Rule.

An inserted implicit conversion must be in scope as a single identifier or be associated with the source or target type of the conversion.

"Associated with" means found in the companion object of.

Describe the style rule for implicit parameters.

Use at least one role-determining name within the type of an implicit parameter.
Bad:
```
(implicit orderer: (T,T) => Boolean)
```
Good:
```
(implicit orderer: T => Ordered[T])
```

Preferred Prompt, not String.

Give the two main uses of implicit receiver conversion.

- Integrating a new class into an existing class hierarchy.
- Creating DSLs, syntax-like extensions. You can't add your new "syntax" (e.g. `->`) to `Any`

Return fixed elements along with variable part using `unapplySeq`.

Return `Option [TupleN]` for some N, since `Tuple` is a `Seq`.

For example, return
`Option[(String,Seq[String])]`