Implement `pixels-from-corner`, which takes a position on the screen and returns the total number of pixels to move down and over to reach the position.
Design Recipe II

**Data**
- Understand the input data

**Contract, Purpose, and Header**
- Describe (but don’t write) the function

**Examples**
- Show what will happen when the function is done

**Template**
- Set up the body based on the input data (and *only* the input)

**Body**
- The most creative step: implement the function body

**Test**
- Run the examples
Design Recipe II

Data

- Understand the input data

Contract, Purpose, and Header

- Describe (but don’t write) the function

Examples

- Show what will happen when the function is done

Template

- Set up the body based on the input data (and only the input)

Body

- The most creative step: implement the function body

Test

- Run the examples

Implement `pixels-from-corner`, which takes a position on the screen and returns the total number of pixels to move down and over to reach the position.
Design Recipe II

Data
- Understand the input data

Contract, Purpose, and Header
- Describe (but don’t write) the function

Examples
- Show what will happen when the function is done

Template
- Set up the body based on the input data (and only the input)

Body
- The most creative

Test
- Run the examples

Implement `pixels-from-corner`, which takes a position on the screen and returns the total number of pixels to move down and over to reach the position
Design Recipe II

Data

• Understand the input data

Contract, Purpose, and Header

• Describe (but don’t write) the function

Examples

• Show what will happen when the function is done

Template

• Set up the body based on the input data (and only the input)

Body

• The most creative step: implement the function body

Test

• Run the examples

Implement *pixels-from-corner*, which takes a position on the screen and returns the total number of pixels to move down and over to reach the position.
Design Recipe II

**Data**
- Understand the input data

**Contract, Purpose, and Header**
- Describe (but don’t write) the function

**Examples**
- Show what will happen when the function is done

**Template**
- Set up the body based on the input data (and *only* the input)

**Body**
- The most creative
  - Implement `pixels-from-corner`, which takes a position on the screen and returns the total number of pixels to move down and over to reach the position

**Test**
- Run the examples
Design Recipe II

**Data**
- Understand the input data

**Contract, Purpose, and Header**
- Describe (but don’t write) the function

**Examples**
- Show what will happen when the function is done

**Template**
- Set up the body based on the input data (and only the input)

**Body**
- The most creative step: implement the function body.

**Test**
- Run the examples

Implement `pixels-from-corner`, which takes a position on the screen and returns the total number of pixels to move down and over to reach the position
# Design Recipe II

## Data
- Understand the input data

## Contract, Purpose, and Header
- Describe (but don’t write) the function

## Examples
- Show what will happen when the function is done

## Template
- Set up the body based on the input data (and only the input)

## Body
- The most creative step: implement the function body

## Test
- Run the examples

Implement `pixels-from-corner`, which takes a position on the screen and returns the total number of pixels to move down and over to reach the position...
Implement \textit{flip-posn}, which takes a position and flips it over the diagonal
# Design Recipe II

## Data
- Understand the input data

## Contract, Purpose, and Header
- Describe (but don’t write) the function

## Examples
- Show what will happen when the function is done

## Template
- Set up the body based on the input data (and only the input)

## Body
- The most creative step: implement the function body

## Test
- Run the examples

Implement `flip-posn`, which takes a position and flips it over the diagonal
Design Recipe II

**Data**
- Understand the input data

**Contract, Purpose, and Header**
- Describe (but don’t write) the function

**Examples**
- Show what will happen when the function is done

**Template**
- Set up the body based on the input data (and only the input)

**Body**
- The most creative step: implement the function body

**Test**
- Run the examples
  Implement `flip-posn`, which takes a position and flips it over the diagonal
Design Recipe II

Data
• Understand the input data

Contract, Purpose, and Header
• Describe (but don’t write) the function

Examples
• Show what will happen when the function is done

Template
• Set up the body based on the input data (and only the input)

Body
• The most creative step: implement the function body

Test
• Run the examples

Implement \texttt{flip-posn}, which takes a position and flips it over the diagonal
Design Recipe II

Data

• Understand the input data

Contract, Purpose, and Header

• Describe (but don’t write) the function

Examples

• Show what will happen when the function is done

Template

• Set up the body based on the input data (and only the input)

Body

• The most creative step: implement the function body

Test

• Run the examples

Implement \texttt{flip-posn}, which takes a position and flips it over the diagonal
Design Recipe II

Data

• Understand the input data

Contract, Purpose, and Header

• Describe (but don’t write) the function

Examples

• Show what will happen when the function is done

Template

• Set up the body based on the input data (and only the input)

Body

• The most creative step: implement the function body

Test

• Run the examples

Implement `flip-posn`, which takes a position and flips it over the diagonal
Design Recipe II

**Data**

- Understand the input data

**Contract, Purpose, and Header**

- Describe (but don’t write) the function

**Examples**

- Show what will happen when the function is done

**Template**

- Set up the body based on the input data (and *only* the input)

**Body**

- The most creative step: implement the function body

**Test**

- Run the examples
  - Implement `flip-posn`, which takes a position and flips it over the diagonal