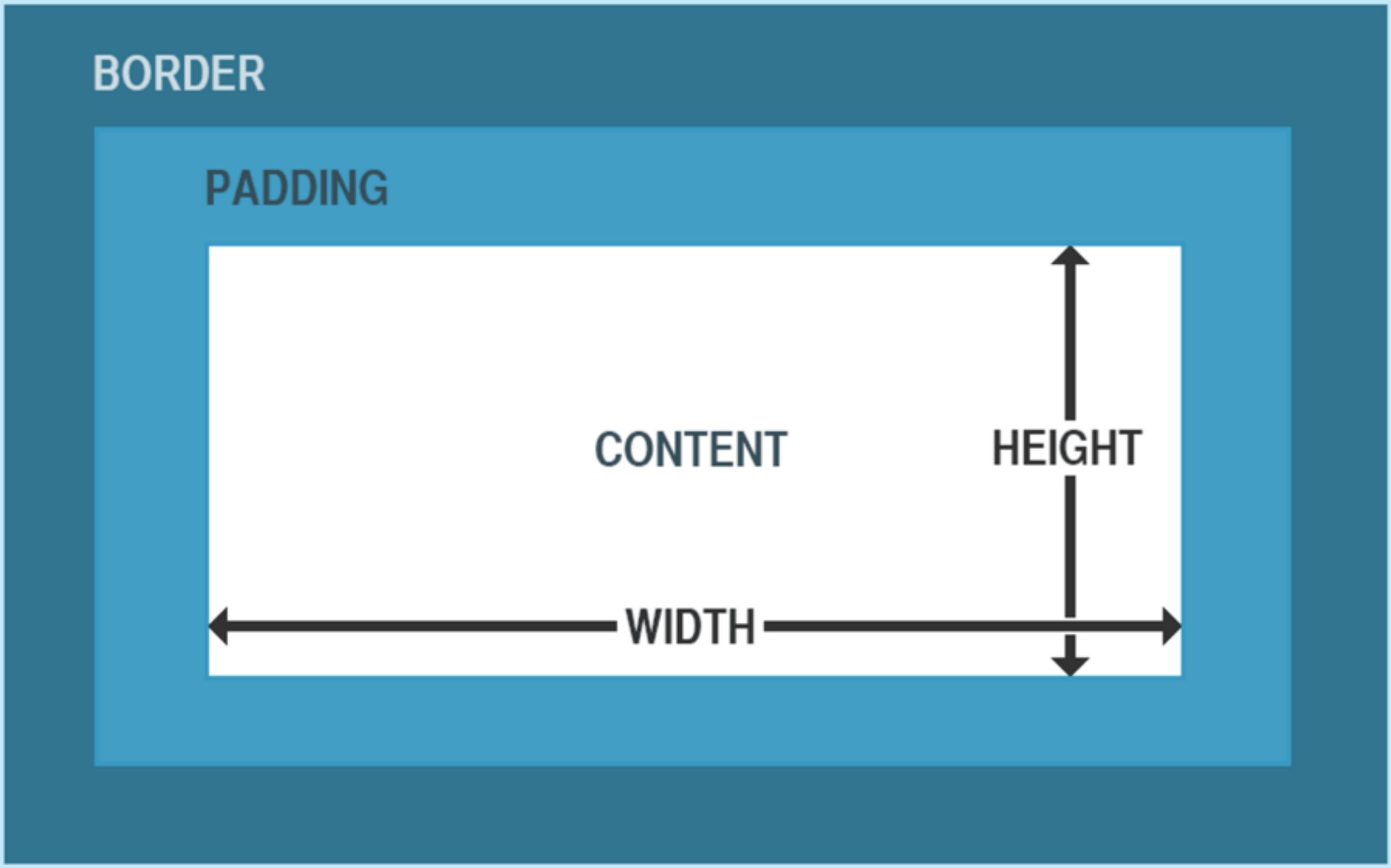
CSS 102

Box Model



http://inserthtml.com/codex/wp-content/uploads/2012/06/box-model1.gif

Box Model Example

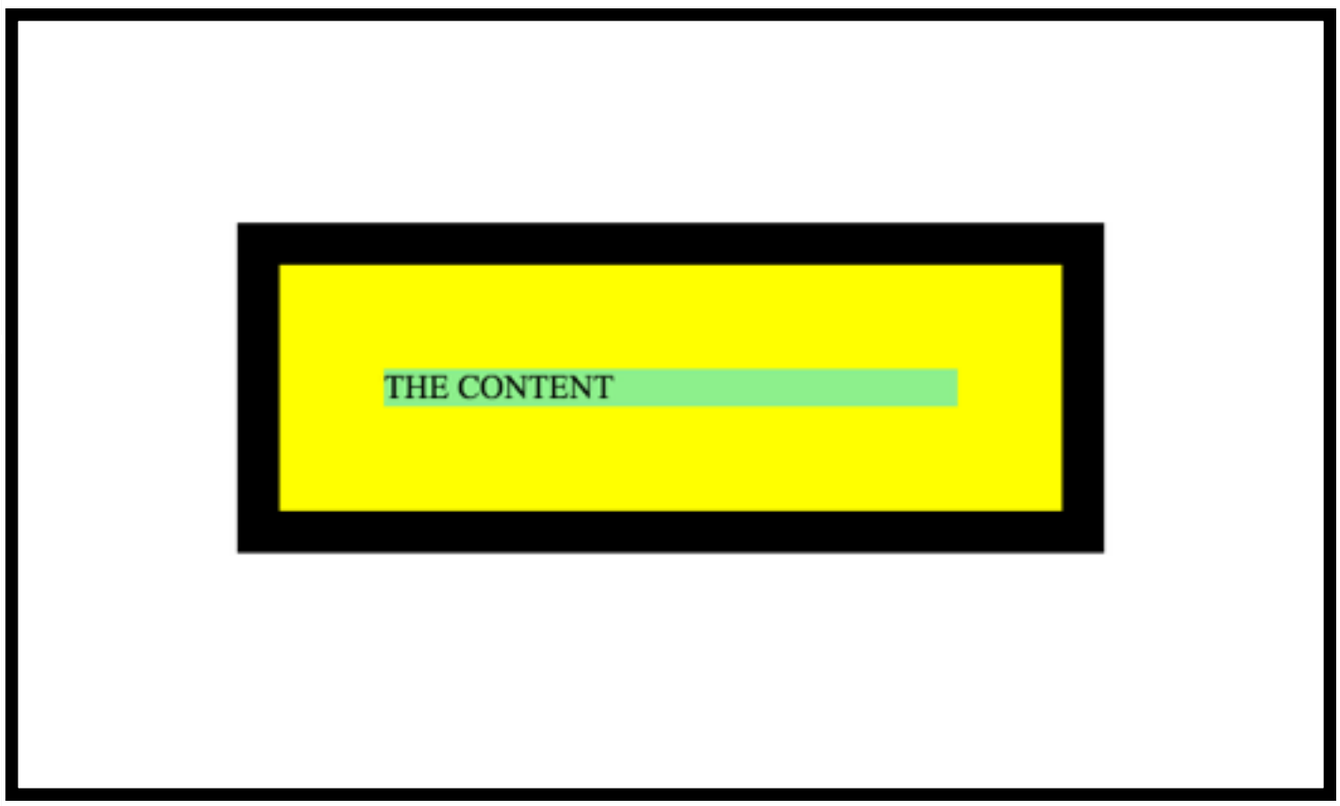
```html
<div class="box">
  THE CONTENT
</div>
```

```css
.box {
  background-color: yellow;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

Box Model Example

```html
<div class="box">
  THE CONTENT
</div>
```

```css
.box {
  background-color: yellow;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

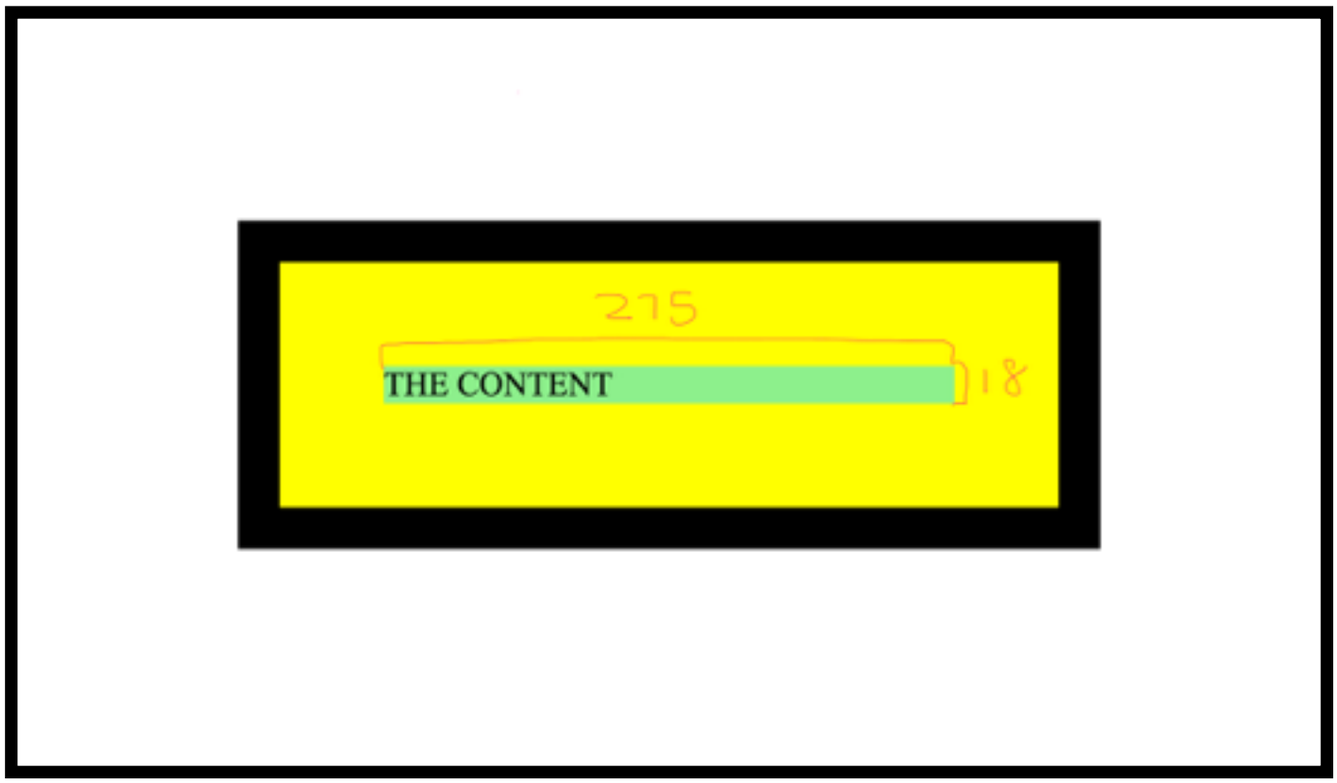


Imagine this fictitious green box around the content, denoting the content box

Box Model Example

```html
<div class="box">
  THE CONTENT
</div>
```

```css
.box {
  background-color: yellow;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



The width of the box stretches to the maximally available horizontal space

Box Model Example

```html
<div class="box">
  THE CONTENT
</div>
```

```css
.box {
  background-color: yellow;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

The height of the content box adjusts to the amount of content

Box Model Example

```html
<div class="box">
  THE CONTENT
</div>
```

```css
.box {
  background-color: yellow;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

The height of the content box adjusts to the amount of content

Padding

```css
.box {
  background-color: yellow;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

Border

```
.box {
  background-color: yellow;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

Margin

```
.box {
  background-color: yellow;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

Specifying Different values for Each Side

- `padding-top, padding-bottom, padding-left, padding-right`

- `border-top, border-bottom, border-left, border-right`

- `margin-top, margin-bottom, margin-left, margin-right`

What if you specific the height and the width?

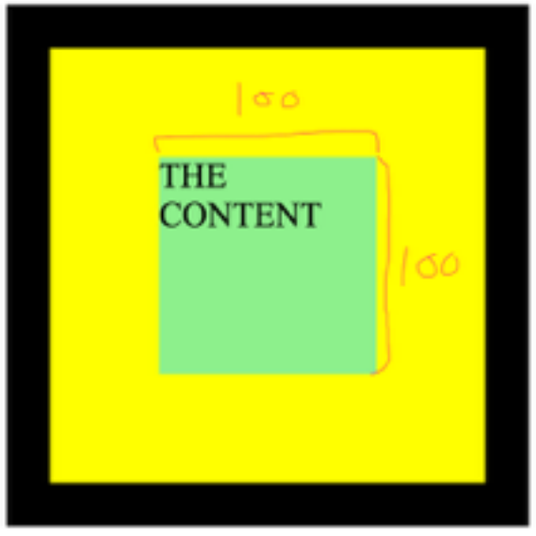
```
.box {
    background-color: yellow;
    height: 100px;
    width: 100px;
    padding: 50px;
    border: 20px solid black;
    margin: 100px;
}
```

What if you specific the height and the width?

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

What if you specific the height and the width?

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



What if you have too much content?

What happens when you have too much content?

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

`overflow: scroll` or `overflow: auto`

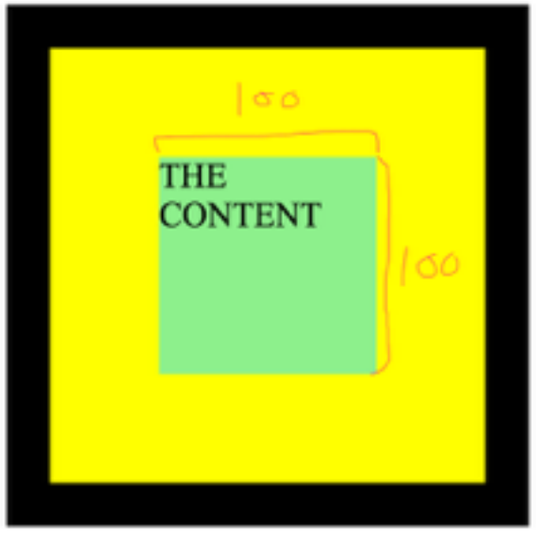
```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
  overflow: scroll;
}
```

overflow: hidden

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
  overflow: hidden;
}
```

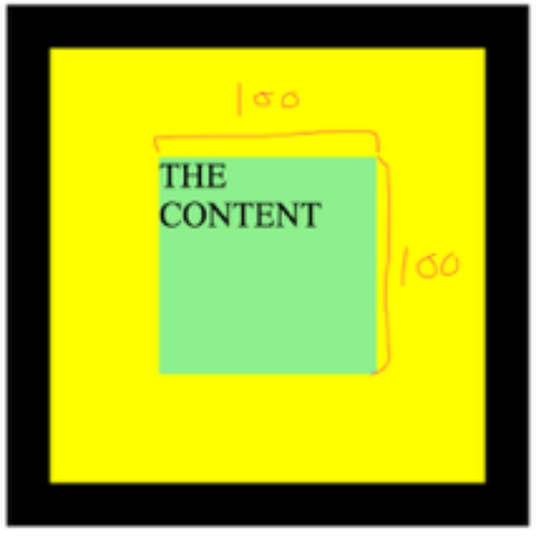
More on height and width

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



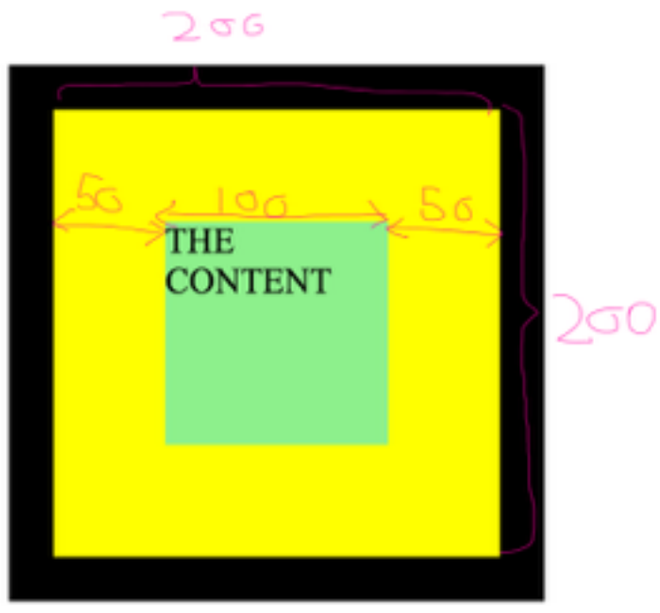
What are the height and width of the yellow box?

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



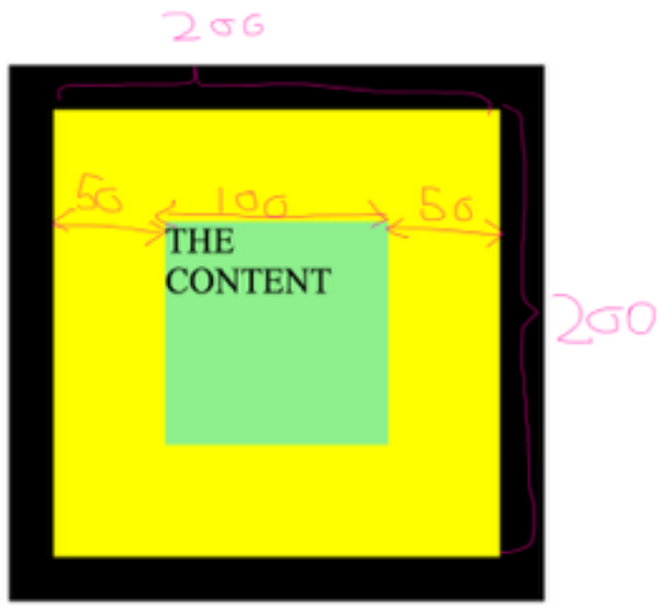
What are the height and width of the yellow box?

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



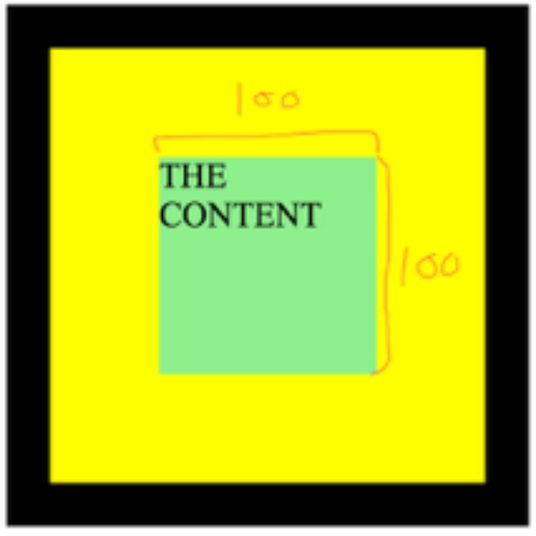
What are the height and width of the yellow box?

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



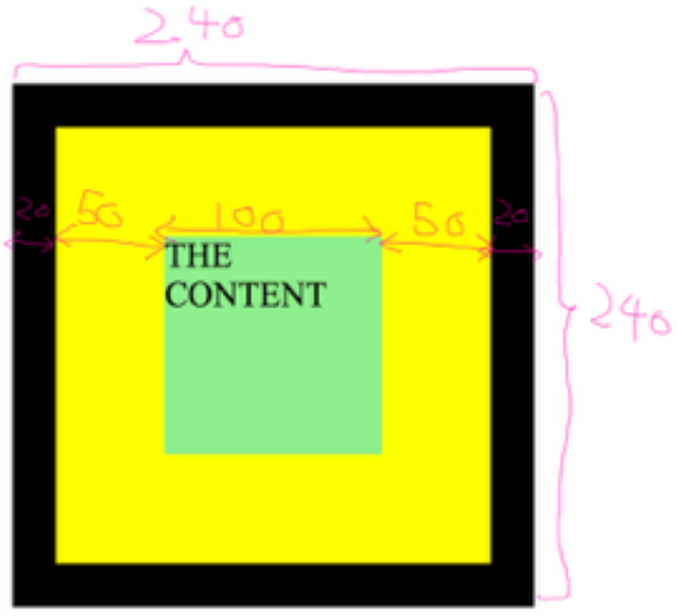
What are the height and width of the black box?

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



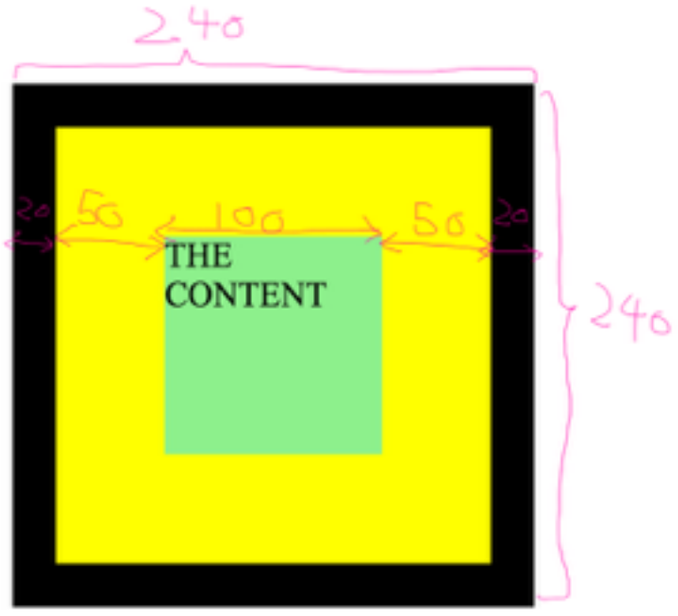
What are the height and width of the black box?

```css
.box {
  background-color: yellow;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



Shouldn’t we specify the size of the border box instead?

```css
.box {
  background-color: yellow;
  height: 240px;
  width: 240px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



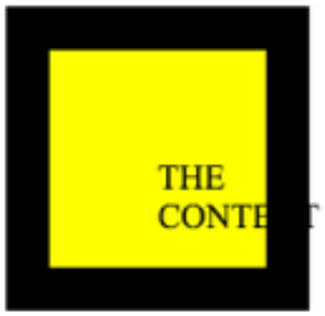
You can do that!

box-sizing: border-box

```css
.box {
  background-color: yellow;
  box-sizing: border-box;
  height: 240px;
  width: 240px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```

box-sizing: border-box

```css
.box {
  background-color: yellow;
  box-sizing: border-box;
  height: 100px;
  width: 100px;
  padding: 50px;
  border: 20px solid black;
  margin: 100px;
}
```



The `display` property

common display property values

- none
- inline
- block
- inline-block
- table-cell
- there are more…

```html
<div>Hello, I am a div.</div>
<span>Hello, I am a span.
<i>Hello</i></span> some more text
<a href="#">Hello, I am an anchor.</a>
```

Hello, I am a div.
Hello, I am a span. *Hello* some more text Hello, I am an anchor.

```
<style>
div {
  background-color: cyan;
  border: 2px solid black;
}

span {
  background-color: yellow;
  border: 2px solid black;
}

a {
  background-color: orange;
  border: 2px solid black;
}
</style>
```

Hello, I am a div.

Hello, I am a span. *Hello* some more text Hello, I am an anchor.

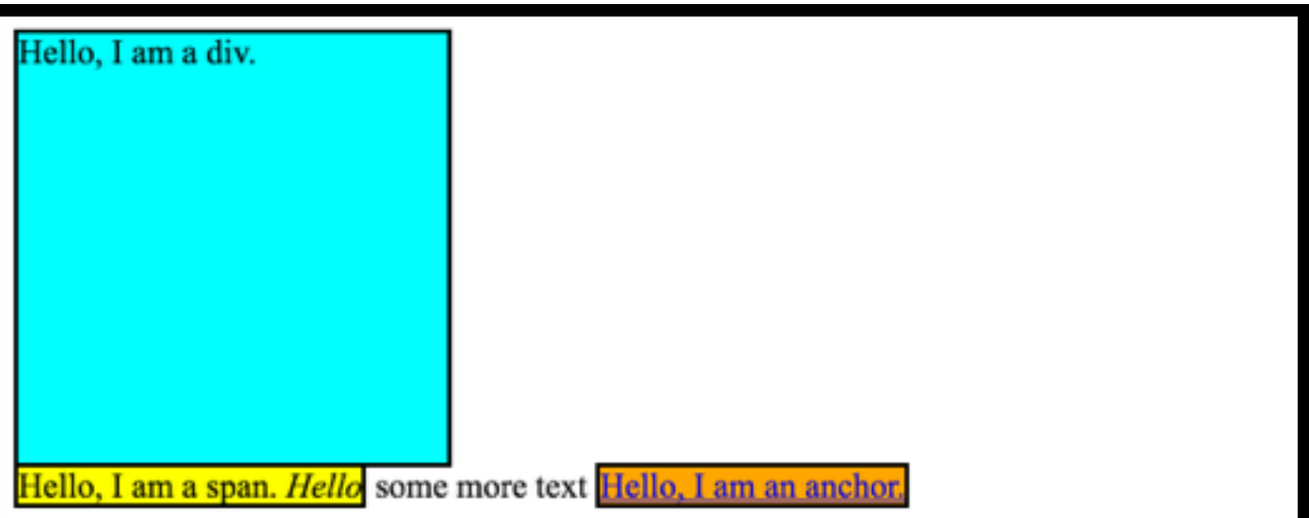
Hello, I am a div.

Hello, I am a span. *Hello* some more text Hello, I am an anchor.

```
<style>
div {
  background-color: cyan;
  border: 2px solid black;
  padding: 10px;
  margin: 20px;
}

span {
  background-color: yellow;
  border: 2px solid black;
  padding: 10px;
  margin: 20px;
}

a {
  background-color: orange;
  border: 2px solid black;
  padding: 10px;
  margin: 20px;
}
</style>
```

Setting Width and Height doesn't work for inlined elements

```
<style>
div {
  background-color: cyan;
  border: 2px solid black;
  width: 200px;
  height: 200px;
}

span {
  background-color: yellow;
  border: 2px solid black;
  width: 200px;
  height: 200px;
}

a {
  background-color: orange;
  border: 2px solid black;
}
</style>
```



Common Block-level Elements

- div
- h1-h5
- ol, ul, li
- p
- form
- blockquote
- hr

Common Inline Elements

- img
- a
- span
- em
- strong
- input, button, label, select, textarea

But you can change the display!

```css
span {
  background-color: yellow;
  border: 2px solid black;
  padding: 10px;
  margin: 20px;
  display: block;
}
```

Hello, I am a div.

Hello, I am a span. *Hello*

some more text Hello, I am an anchor.

inline-block

inline-block elements

- like inline
- can have height and width

Images are inline-block

```
<div>Hello, I am a div.</div>
<span>Hello, I am a span.
<i>Hello</i></span> some more text
<a href="#">Hello, I am an anchor.</a>
<img src="lolcat1.jpg" alt="lolcat">
```

Hello, I am a div.

Hello, I am a span. *Hello* some more text Hello, I am an anchor.

You can make things inline-block

```
.inline-block-element {
  display: inline-block;
  width: 200px;
  height: 200px;
  background-color: lightpink;
  vertical-align: top;
  margin: 0
}
</style>
</head>
<body>
<div>Hello, I am a div.</div>
<div class="inline-block-element">
  <span>Hello, I am a span.
  <i>Hello</i></span> some more text
  <a href="#">Hello, I am an anchor.</a>
</div>
<img src="lolcat1.jpg" alt="lolcat">
```

Hello, I am a div.

Hello, I am a span. *Hello* some more text Hello, I am an anchor.

Centering Stuff

The Google Search Page

Google Search I'm Feeling Lucky

This Earth Day, explore stories from our beautiful planet with Google Maps

text-align: center

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Google</title>
    <style>
    body {
      text-align: center;
    }
    </style>
  </head>
  <body>
    <img src="https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png"
    alt="Google"><br>
    <input type="text" name="q"><br>
    <button>Google Search</button>
    <button>I'm Feeling Lucky</button>
    <p>This Earth Day, explore stories from our beautiful planet with Google Maps</p>
  </body>
</html>
```

A Blog Layout

April 21, 2011

A new micro clearfix

The clearfix hack is a popular way to contain floats without resorting to using presentational markup. This article presents an update to the clearfix method that further reduces the amount of CSS required.

Demo: Micro clearfix hack

Known support: Firefox 3.5+, Safari 4+, Chrome, Opera 9+, IE 6+

The "micro clearfix" method is suitable for modern browsers and builds upon Thierry Koblentz's "clearfix reloaded", which introduced the use of both the :before and :after pseudo-elements.

Here is the updated code (I've used a shorter class name too):

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug
 *    contenteditable attribute is included anywhere els
 *    Otherwise it causes space to appear at the top and
 *    that are clearfixed.
 * 2. The use of `table` rather than `block` is only nec
 *    `:before` to contain the top-margins of child elem
 */
.cf:before,
.cf:after {
    content: " "; /* 1 */
    display: table; /* 2 */
}

.cf:after {
    clear: both;
}

/**
 * For IE 6/7 only
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Blog</title>
    <style>
    body { background: gray; }
    pre { overflow: scroll; }

    #content {
      width: 500px;
      background-color: white;
      padding: 20px;
    }
    </style>
  </head>
  <body>
    <div id="content">…
  </body>
</html>
```

April 21, 2011

A new micro clearfix

The clearfix hack is a popular way to contain floats without resorting to using presentational markup. This article presents an update to the clearfix method that further reduces the amount of CSS required.

Demo: Micro clearfix hack

Known support: Firefox 3.5+, Safari 4+, Chrome, Opera 9+, IE 6+

The “micro clearfix” method is suitable for modern browsers and builds upon Thierry Koblentz’s “clearfix reloaded”, which introduced the use of both the :before and :after pseudo-elements.

Here is the updated code (I’ve used a shorter class name too):

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug
 *    contenteditable attribute is included anywhere els
 *    Otherwise it causes space to appear at the top and
 *    that are clearfixed.
 * 2. The use of `table` rather than `block` is only nec
 *    `:before` to contain the top-margins of child elem
 */
.cf:before,
.cf:after {
    content: " "; /* 1 */
    display: table; /* 2 */
}

.cf:after {
    clear: both;
}
```

text-align: center

That didn’t work

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Blog</title>
    <style>
    body { background: gray; }
    pre { overflow: scroll; }

    #content {
      width: 500px;
      background-color: white;
      padding: 20px;
      text-align: center;
    }
    </style>
  </head>
  <body>
    <div id="content">
  </body>
</html>
```

April 21, 2011

A new micro clearfix

The clearfix hack is a popular way to contain floats without resorting to using presentational markup. This article presents an update to the clearfix method that further reduces the amount of CSS required.

Demo: Micro clearfix hack

Known support: Firefox 3.5+, Safari 4+, Chrome, Opera 9+, IE 6+

The “micro clearfix” method is suitable for modern browsers and builds upon Thierry Koblentz’s “clearfix reloaded”, which introduced the use of both the :before and :after pseudo-elements.

Here is the updated code (I’ve used a shorter class name too):

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug
 *    contenteditable attribute is included anywhere els
 *    Otherwise it causes space to appear at the top and
 *    that are clearfixed.
 * 2. The use of `table` rather than `block` is only nec
 *    `:before` to contain the top-margins of child elem
 */
.cf:before,
.cf:after {
  content: " "; /* 1 */
  display: table; /* 2 */
}

.cf:after {
  clear: both;
}
```

margin: auto

margin-left: auto pushes to the right

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Blog</title>
    <style>
    body { background: gray; }
    pre { overflow: scroll; }

    #content {
      width: 500px;
      background-color: white;
      padding: 20px;
      margin-left: auto;
    }
    </style>
  </head>
  <body>
    <div id="content">
  </body>
</html>
```

April 21, 2011

A new micro clearfix

The clearfix hack is a popular way to contain floats without resorting to using presentational markup. This article presents an update to the clearfix method that further reduces the amount of CSS required.

Demo: Micro clearfix hack

Known support: Firefox 3.5+, Safari 4+, Chrome, Opera 9+, IE 6+

The “micro clearfix” method is suitable for modern browsers and builds upon Thierry Koblentz’s “clearfix reloaded”, which introduced the use of both the :before and :after pseudo-elements.

Here is the updated code (I’ve used a shorter class name too):

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug
 *    contenteditable attribute is included anywhere els
 *    Otherwise it causes space to appear at the top and
 *    that are clearfixed.
 * 2. The use of `table` rather than `block` is only nec
 *    `:before` to contain the top-margins of child elem
 */
.cf:before,
.cf:after {
    content: " "; /* 1 */
    display: table; /* 2 */
}

.cf:after {
    clear: both;
}
```

margin-right: auto pushes to the left

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Blog</title>
    <style>
    body { background: gray; }
    pre { overflow: scroll; }

    #content {
      width: 500px;
      background-color: white;
      padding: 20px;
      margin-right: auto;
    }
    </style>
  </head>
  <body>
    <div id="content">
  </body>
</html>
```

April 21, 2011

A new micro clearfix

The clearfix hack is a popular way to contain floats without resorting to using presentational markup. This article presents an update to the clearfix method that further reduces the amount of CSS required.

Demo: Micro clearfix hack

Known support: Firefox 3.5+, Safari 4+, Chrome, Opera 9+, IE 6+

The “micro clearfix” method is suitable for modern browsers and builds upon Thierry Koblentz’s “clearfix reloaded”, which introduced the use of both the :before and :after pseudo-elements.

Here is the updated code (I’ve used a shorter class name too):

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug
 *    contenteditable attribute is included anywhere els
 *    Otherwise it causes space to appear at the top and
 *    that are clearfixed.
 * 2. The use of `table` rather than `block` is only nec
 *    `:before` to contain the top-margins of child elem
 */
.cf:before,
.cf:after {
    content: " "; /* 1 */
    display: table; /* 2 */
}

.cf:after {
    clear: both;
}
```

Combining them pushes to the center

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Blog</title>
    <style>
    body { background: gray; }
    pre { overflow: scroll; }

    #content {
      width: 500px;
      background-color: white;
      padding: 20px;
      margin-left: auto;
      margin-right: auto;
    }
    </style>
  </head>
  <body>
    <div id="content">…
  </body>
</html>
```

April 21, 2011

A new micro clearfix

The clearfix hack is a popular way to contain floats without resorting to using presentational markup. This article presents an update to the clearfix method that further reduces the amount of CSS required.

Demo: Micro clearfix hack

Known support: Firefox 3.5+, Safari 4+, Chrome, Opera 9+, IE 6+

The “micro clearfix” method is suitable for modern browsers and builds upon Thierry Koblentz’s “clearfix reloaded”, which introduced the use of both the :before and :after pseudo-elements.

Here is the updated code (I’ve used a shorter class name too):

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug
 *    contenteditable attribute is included anywhere els
 *    Otherwise it causes space to appear at the top and
 *    that are clearfixed.
 * 2. The use of `table` rather than `block` is only nec
 *    `:before` to contain the top-margins of child elem
 */
.cf:before,
.cf:after {
    content: " "; /* 1 */
    display: table; /* 2 */
}

.cf:after {
    clear: both;
}

/**
 * For IE 6/7 only
```

Condensed Version

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Blog</title>
    <style>
    body { background: gray; }
    pre { overflow: scroll; }

    #content {
      width: 500px;
      background-color: white;
      padding: 20px;
      margin: auto;
    }
    </style>
  </head>
  <body>
    <div id="content">
  </body>
</html>
```

April 21, 2011

A new micro clearfix

The clearfix hack is a popular way to contain floats without resorting to using presentational markup. This article presents an update to the clearfix method that further reduces the amount of CSS required.

Demo: Micro clearfix hack

Known support: Firefox 3.5+, Safari 4+, Chrome, Opera 9+, IE 6+

The “micro clearfix” method is suitable for modern browsers and builds upon Thierry Koblentz’s “clearfix reloaded”, which introduced the use of both the :before and :after pseudo-elements.

Here is the updated code (I’ve used a shorter class name too):

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug
 *    contenteditable attribute is included anywhere els
 *    Otherwise it causes space to appear at the top and
 *    that are clearfixed.
 * 2. The use of `table` rather than `block` is only nec
 *    `:before` to contain the top-margins of child elem
 */
.cf:before,
.cf:after {
    content: " "; /* 1 */
    display: table; /* 2 */
}

.cf:after {
    clear: both;
}
```