Strings

HFCC – CIS222

20 & 22 February 2012

Announcements:

- ✔ Quiz 6 (Array) on Wednesday (22 Feb)
- ✔ Homework 4 (Arrays) on Monday (27 Feb)
- ✔ Midterm on 29 Feb
  - Review on Monday the 27th
  - That's next week, y'all!!
- ✔ Project Part 1 due on 29 Feb
  - At the start of the midterm

Strings

- Strings are just arrays of characters
  - Characters are any ASCII character
    - No Unicode!
    - A-Z, a-z, 0-9, special characters
    - Even hard-to-type special characters
      a. Newline (\n)
      b. Tab (\t)
    - And reserved special characters
      a. Dollar sign (\$)
      b. Double quote (\")
      c. Backslash (\\)
- No limit on size (aside from system memory)

Storing Strings

- Double Quotes
  - Parses special characters and variables

```
$myName = “Jer”;
$sentence = “$myName is awesome!”;     // Contains Jer is awesome!
$sentence = “$sentence \n$sentence”;
```

- Single Quotes
  - Does no parsing of special characters or variables
    - Except you can use \' to display a single quote and \\ for a backslash

```
$myName = 'Jer';
$sentence = '$myName is awesome!”;     // Contains $myName is awesome!
```

Storing Strings

- Heredoc
  - Use a delimiter on both sides of the string to store larger strings
    - Delimiter can be any string
    - Starts with **<<<delimiter**
    - Ends with **delimiter;** all alone at the start of a line
  - Parses special characters and variables

```
$myBio = <<<TEXT
    Here is some text that is all about "$myName" and the things
    that "$myName" likes!
TEXT;
```

  - $myName contains:

Here is some text that is all about "Jer" and the things that "Jer" likes!

  - No need to escape quotes!

Storing Strings

Nowdoc

- Exactly like Heredoc, except does not parse variables or special characters
  - It's essentially the single quote version of Heredoc
  - Enclose the opening delimiter in single quotes

```
$myBio = <<<'TEXT'
    Here is some text that is all about “$myName” and the things
    that “$myName” likes!
TEXT;
```

- $myName contains:

Here is some text that is all about “$myName” and the things that “$myName” likes!

Concatenation

- Join two strings with a .

```
$myBio = 'string 1'.'string 2';
```

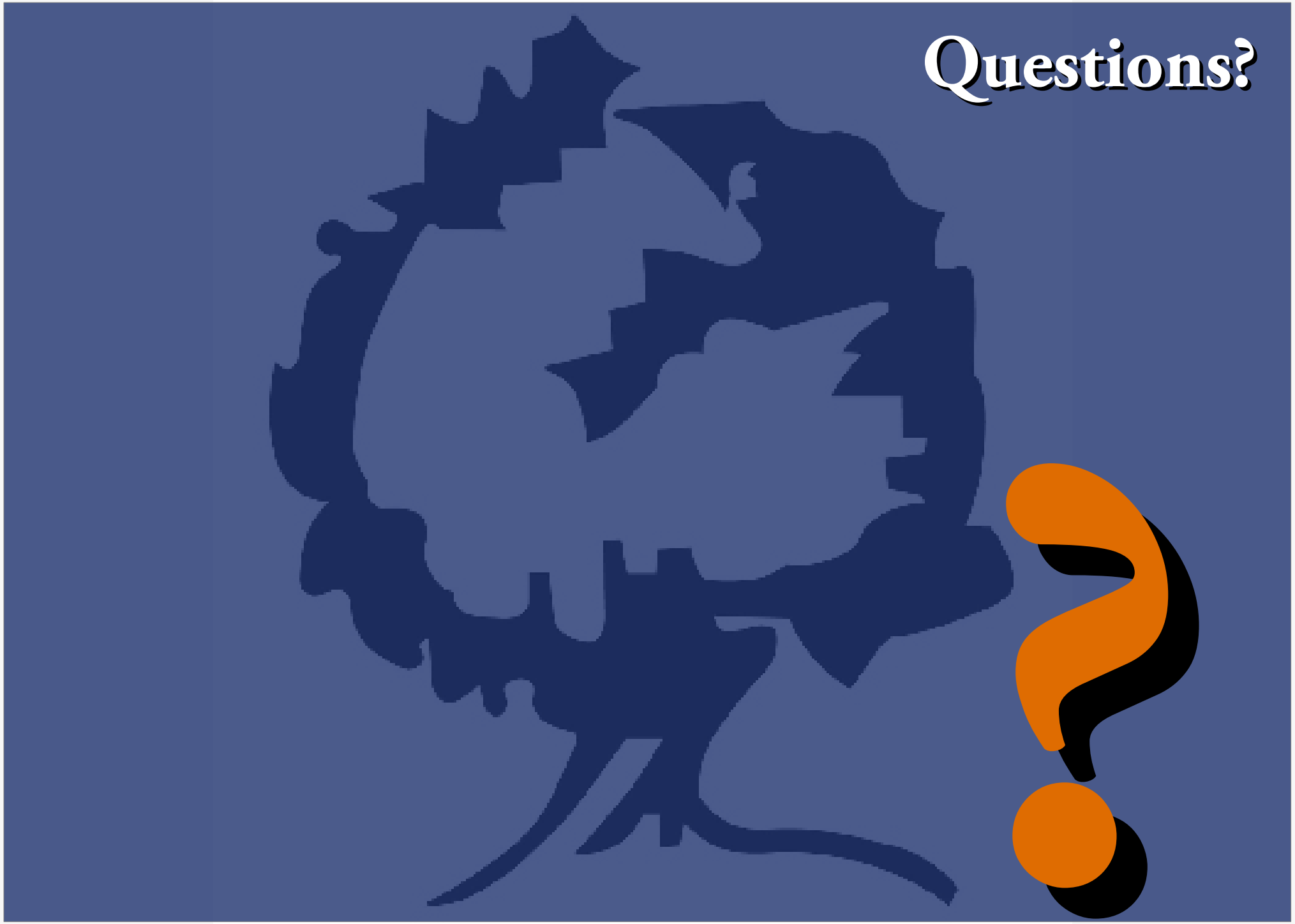
- Note: be careful with decimals...

```
$num = 1.2;     // Contains float 1.2
$num = 1 . 2;     // Contains string 12
```

- Can also use as an assignment operator

```
$sentence = "The quick brown foxes";
$sentence .= " jumped over the lazy dogs!";
// Contains: The quick brown foxes jumped over the lazy dogs!
```

Questions?



Strings as Arrays

- Strings are arrays of characters
  - That's right, I said **array**

```
$myName = “Jeremy”;   // Contains Jeremy
$myGrade = “76.5”;   // Contains 76.5
echo $myName[2];     // Outputs r
echo $myGrade[1];     // Outputs 6
```

- In strings, we call the index the **position**
  - But you can still call it the index if you want
  - I won't tell

String Functions

- **strlen($string);**
  - Returns the number of characters in the string
    - Like the **count()** function of an array

```
echo strlen("Jeremy");    // Outputs 6
```

- **strpos($string, $stringToFind);**
  - Returns the **position** of the first occurrance of $stringToFind in $string
  - Returns *false* if $stringToFind is not found
    - You must use === comparison, because the position could be 0!
  - Optionally, can add an integer offset to skip some positions
    - Offset is **counted**, not a **position**

```
echo strpos("Jeremy", "e");     // Outputs 1
echo strpos("Jeremy", "e", 2);    // Outputs 3
echo strpos("Jeremy", "q");     // Outputs false
```

String Functions

- **strtoupper($string);** and **strtolower($string);**
  - Returns the string in all upper case (or all lower case)
    - Does not affect the actual string!

```
$myName = "Jeremy";
echo strtoupper($myName);    // Outputs JEREMY
echo $myName;     // Outputs Jeremy
$myName = $strtoupper($myName);
echo $myName;     // Outputs JEREMY
```

String Functions

substr($string, $startPosition, *$charCount*);

- Returns part of $string starting at $startPosition and including $charCount characters
  - $startPosition is **position** or **index** based
  - $charCount is **count** based

```
echo substr("Jeremy", 0,3);     // Outputs Jer
echo substr("Jeremy", 2, 3);    // Outputs rem
```

- $charCount is optional
  - Without $charCount, the rest of the string is returned

```
echo substr("Jeremy", 2);     // Outputs remy
```

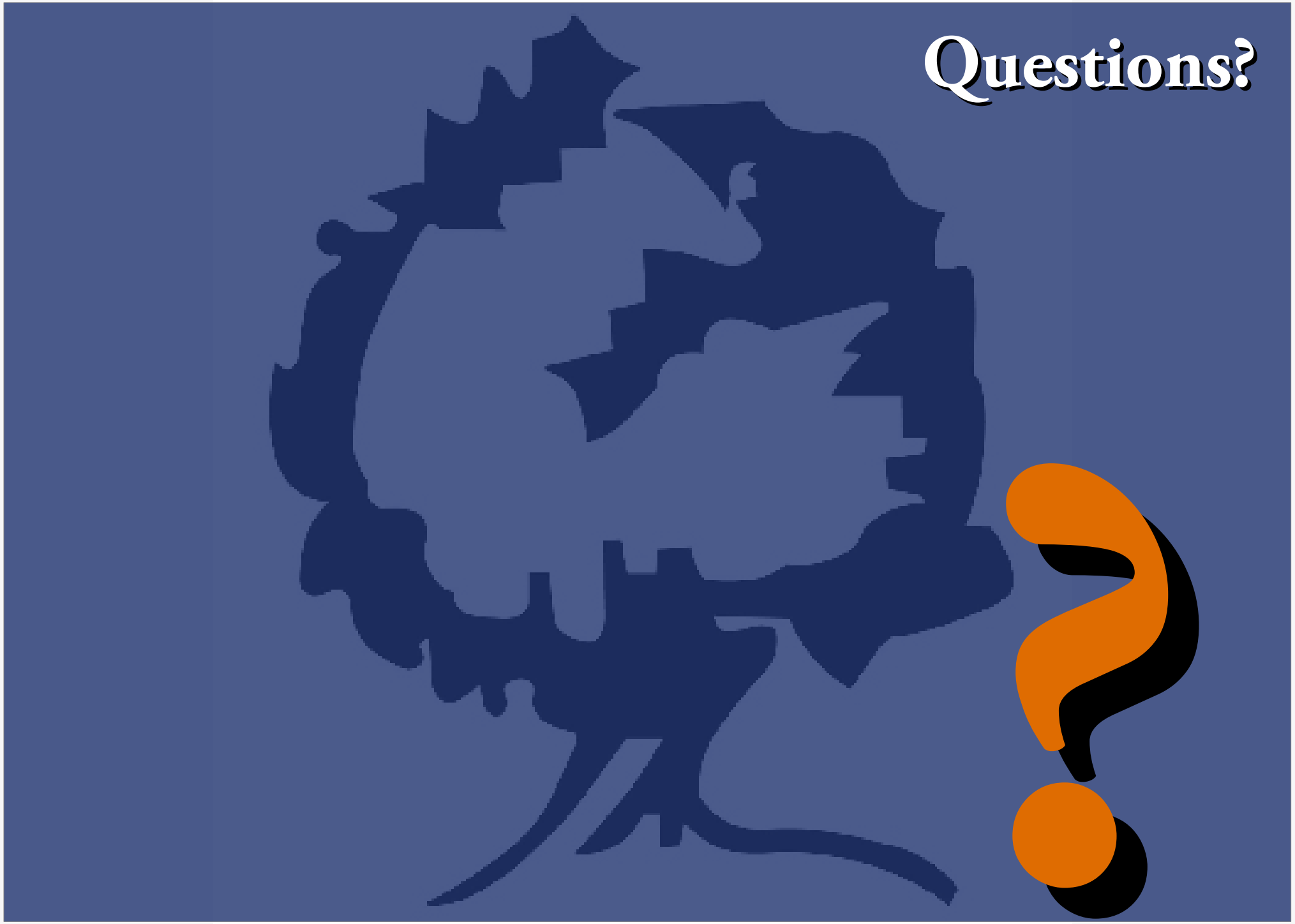
- $startPosition can be negative
  - Counts its position from the end, then

```
echo substr("Jeremy", -3, 3);     // Outputs emy
echo substr("Jeremy", -4);     // Outputs remy
```

- $charCount can be negative
  - It means to omit $that many characters from the end

```
echo substr("Jeremy", 1,-2);     // Outputs ere
echo substr("Jeremy", -5, -2);     // Also outputs ere
```

Questions?



Let's Try It!

$str =	J	e	r		i	s		a	w	e	s	o	m	e	!
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
echo strlen($str);

echo strpos($str, “s”);

echo strpos($str, “s”, 10);

echo substr($str, 4, 2);

echo substr($str, -5, 4);

echo substr($str, -8, -1);
```

Let's Try It!

```
$str = “Every rose has its thorn, just like every night, has its dawn”;

echo strlen($str);

echo strpos($str, “its”, 14);

echo substr($str, -19);

$str2 = “. Just like every cowboy, sings a sad, sad song.”;
$str .= $str2.' '.$str;

echo $str;
```

String Functions

- **trim($string);**
  - Returns the string with all whitespace cut off the head and the tail
    - Whitespace, by default, includes: spaces, tabs, newlines, null bytes, etc

```
echo trim(“ Jeremy     ”);     // Outputs Jeremy
```

    - Optionally, you can add a second argument as a string list of characters to trim

```
echo trim(“ Jeremy   ”, “ ey”);    // Outputs Jerem
```

- **htmlentities($string);**
  - Returns the string with all characters that have an HTML entity code translated to those codes

```
echo htmlentities(“My name is <b>Jer</b>”);
// Outputs My name is &lt;b&gt;Jer&lt;/b&gt;
```

String Functions

- **substr_replace($string, $insert, $startPosition**, *$length*);
  - Returns $string with $insert replacing the characters that appear starting at $startPosition
    - Default, replaces the same number of characters as the length of $insert or until end of $string
      - Whichever is smaller

```
echo substr_replace("Jeremy", "k", 3);     // Outputs Jerkmy
echo substr_replace("Jeremy", "amiah", 3);     // Outputs Jeramiah
```

    - Optionally, you can add a fourth argument to say how many characters to replace

```
echo substr_replace("Jeremy", "k", 3, 3);     // Outputs Jerk
```

    - You can use 0 for the fourth argument to insert $insert without replacement

```
echo substr_replace("Jeremy", "k", 3, 0);     // Outputs Jerkemy
```

(ahh, my old elementary school nickname)

String Functions

- **explode($delimiter, $string);**
  - Splits up $string at all occurrences of $delimiter and returns it as an array

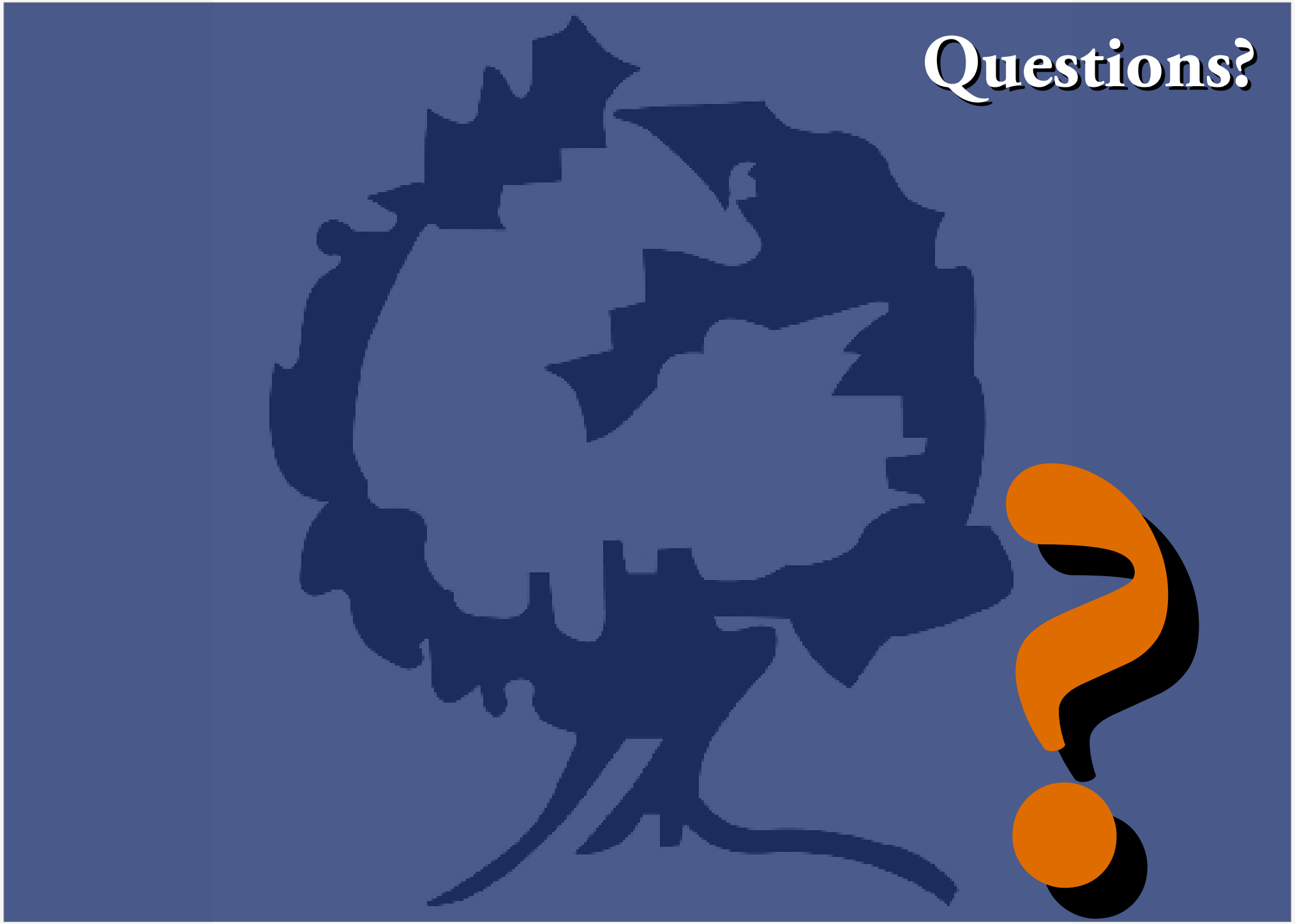
```
$newArr = explode(",", "Jeremy, Stacy, Sally, Sue");
// $newArr is now an array: [0] = Jeremy, [1] = Stacy, [2] = Sally, etc
```

- **implode($delimiter, $array);**
  - Assembles elements in $array separated by $delimiter, returns as a string

```
$arr = array("Jer", "Stacy", "Sally", "Sue");
$newString = implode(":", $arr);
```

$newString contains *Jer:Stacy:Sally:Sue*

Questions?



Let's Try It!

- **Playing with Strings**
- Create a basic page ***InClass-Strings.php***
  - Go to the web address provided in class to get a quote and instructions

    http://cis.hfcc.edu/~jklance/cis222/sandbox/lecture08-Strings/stringEncrypt.php
    - (Probably)
  - Write code that will decode the string according to the instructions provided.
  - Use outputs along the way to see what is happening.
- **Raise your hand if you have finished (or if you are stuck)**

Next Lecture

- Midterm Review
  - Bring your questions!