CS-105
Cryptogram Game

Instructor: Joel Castellanos
e-mail: joel@unm.edu
Web: http://cs.unm.edu/~joel/
Office: Farris Engineering Center (FEC) room 319

3/30/2017

Quiz: Which Image is Drawn?

//Assume the canvas is 200x200 pixels
var i = 0;
while (i<=200)
{
    line(0, 0, 200, i); //x1, y1, x2, y2
    i += 10;
}

a)  b)  c)  d)
Cryptogram Game

- Code Cracker is a one player game.
- The computer chooses a plain text sentence from a database.
- The computer then generates a random mapping of plain text letters to cypher letters.
- The computer then displays the encrypted sentence by replacing each plain text letter with an encrypted letter.
- Spaces and punctuation marks are left unencrypted.
- The player makes a guess in the form of $z = e$.
- If the guess is correct, the computer replaces each instance of the encrypted letter with each instance of the unencrypted letter.

Caesar Cipher

The Caesar Cipher is named after Julius Caesar, who used it with a wrap around shift of 3, to protect messages of military significance.

If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely $D$, for $A$, and so with the others.

—Suetonius, Life of Julius Caesar 56
Substitution Cipher

- A substitution cipher is a method of encryption by which units of plaintext are replaced with ciphertext according to fixed, bijection.

- The bijection (one-to-one onto map) is created as a random mapping..

- Example: Clear Text: HELLO WORLD!
  Encrypted Text: zquuw swxuj!

Additional Cipher Table:

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| n | f | d | j | q | y | a | z | k | p | c | u | t | g | w | m | b | x | h | o | v | i | s | r | l | e |

Guessing Process

- Given a random map and the encrypted text: zquuw swxuj!

- The player guesses: u = L. Then the game displays: zqLLw swxLj!

- Next the player guesses: z = H. Then the game displays: HqLLw swxLj!

- etc.
Lab Requirements: Clear Text

- Start with the provided CryptogramGame.html.
- This program has an input field for the text to be encrypted. Your program must NOT take user input for the clear text. Your program must include at least 20 messages in an array of strings. When your page loads, your program must randomly choose one of the messages.
- Your 20+ messages must have a unifying theme: Quotes from a book or film series, quotes from famous soccer players or politicians or jokes, ...
- Each message must be at least one complete sentence and must be twelve to thirty words long.
- Use only lower-case (no capitals) in your clear text.

Lab Requirements: Cypher Text (1 of 2)

- The given program builds an alphabet array. Then copies that array, then builds a map by randomly shuffling the letters in the map array. That code may used unchanged.
- The given program uses the canvas to display each letter of the clear text, each step in encrypting each letter and the final encrypted letter.
- Your program must NOT display the clear text nor the steps of encryption. Your program must write only the encrypted message to an HTML label with word-wrap enabled. Note: the example program has two word-wrap labels: one used for the map and one for the alphabet. Since your program will display neither the map nor the alphabet, use one of these labels for the encrypted message and delete the other.
Lab Requirements: Cypher Text (2 of 2)

- The given program encrypts each letter of the message, but just displays it in square brackets. Your program needs to build a new totally encrypted message. One way to do this is to create a new variable equal to an empty string:

\[
\text{var cypher} = \"\";
\]

- Then in a loop, like the given loop that prints each encrypted letter, append each encrypted letter to `cypher`.

- As the player makes correct guesses, replace the lower-case encrypted letters in `cypher` with upper-case clear-text letters.

Lab Requirements: Punctuation

- The given program copies spaces and exclamation points as clear text into the encrypted message.

- Your program must do the same and additionally leave commas, periods, question marks, colons, semicolons, apostrophe and hyphens unencrypted.
Lab Requirements: Player Guesses

- The given program has an HTML text input field into which the user can type a clear text word. The JavaScript code reads the input field and encrypts each letter.

- Your program does not take user input for the clear text message, but does need to have two text input field in order to get the player's guess.

- To make a guess, the player:
  1) Enters a single encrypted letter in one input field.
  2) Enters a single letter in the second input field that the player is guessing the entered encrypted letter represents.
  3) Clicks a "Make Guess" button.

Lab Requirements: Guess Results

- When the player guesses correctly, your program must
  1) Traverse (loop through each letter) of the encrypted message and replace each occurrence of the correctly guessed encrypted letter with its corresponding clear text letter in upper-case.
  2) Update the HTML label displaying the updated message.
  3) Draw a happy animation in the canvas.

- When the player guesses incorrectly, your program must use the canvas to indicate its disappointment.

- When the guess is invalid (one or both of the input fields is blank or contains a non-letter or an upper-case letter), your program must use the canvas to indicate its frustration.
Lab Requirements: Game Over

- When all the letters in the message are upper-case, then the message is fully decrypted and the player has won the game. Your program must recognize when this happens and display a happy animation in the canvas.

- There is not a requirement for a lose condition, however, if you want, you may add a maximum number of guesses after which if the player has not won, then he or she loses.

- Your program may, if you want, display the total number of guesses and/or a history of all guesses.

Lab Requirements: Game Instructions

- Your game must be uploaded to a website. Submit the URL to the game in Blackboard Learn.

- Your webpage must include instructions explaining how to play the game.

- Your website must have a logical layout when viewed in a reasonably sized browser window. Including text labels explaining input fields (for example: "Enter an encrypted letter here:").

- Your page must include at least one static .png or .jpg image - not in the canvas, just in the regular HTML as in the given example.
Grading Rubric: 20 Points Total

[3 points]: When loaded, the website randomly chooses between at least 20 themed sentences.

[5 points]: The website correctly displays an encrypted sentence with spaces and punctuation left clear text.

[6 points]: When the player enters a correct guess, the website displays the sentence with the all correctly guessed letters replaced by upper-case clear text.

[2 points]: When the player enters a correct guess a happy something displays in the canvas.

[2 points]: When the player enters an incorrect guess or invalid values, an unhappy something is displayed on the canvas.

[2 points]: When the player wins, a happy animation is displayed in the canvas.

JavaScript String Function: `toUpperCase()`

- JavaScript Strings have a function:

  `
  toUpperCase()
  `

- This returns a new string equal to the original string with all lower case letters converted to upper case.

- Example:

  ```javascript
  var msg = "g";
  msg = msg.toUpperCase();
  console.log(msg);
  ```
JavaScript String's `.charCodeAt(idx)`

```javascript
var word = "a";
console.log(word + ": " + word.charCodeAt(0));

word = "cat";
console.log(word + ": " + word.charCodeAt(0));
```

For a full list of ASCII codes see: [http://www.ascii-code.com/](http://www.ascii-code.com/)

---

JavaScript String Function: `replace(findString, replaceWithString)`

- JavaScript Strings have a function:
  
  ```javascript
  replace(findString, replaceWithString)
  ```

- This returns a new string equal to the original string with the first occurrence of `findString` replaced by `replaceWithString`.

- Example:

  ```javascript
  var msg = "t wcyaaym zb s aztq.";
  msg = msg.replace("a", "P");
  console.log(msg);
  ```

  `t wcYPaym zb s aztq.`
replaceAll helper Function:
\[\text{replaceAll}(str, \text{findStr}, replaceWith)\]

//============================================
// Returns a new string with all occurrences of findStr in str
// replaced with replaceWithStr.
//============================================
function replaceAll(str, findStr, replaceWith)
{
    while (str.indexOf(findStr) >= 0)
    {
        str = str.replace(findStr, replaceWith);
    }
    return str;
}

isLetter helper Function:
\[\text{isLetter}(str)\]

//============================================
// Returns true if and only if the give string is a single,
// lower-case letter.
//============================================
function isLetter(str)
{
    if (str.length != 1) return false;
    if (str.charCodeAt(0) < 'a'.charCodeAt(0))
    { return false; }
    if (str.charCodeAt(0) > 'z'.charCodeAt(0))
    { return false; }
    return true;
}
How to Check a Player Guess

- Let $\text{guessPart1}$ be the character a player entered as the first part of a guess (the encrypted letter).
- Let $\text{guessPart2}$ be the character a player entered as the second part of a guess (the letter he or she thinks the encrypted letter represents in plain text).
- Then, use the map array to encrypt $\text{guessPart2}$. The guess is correct if and only if the result is equal to $\text{guessPart1}$.

Getting it to Work

- Break the assignment into small parts and get each part working before starting work on the next part.
- Place a `console.log(string)` at the start of every function. The `string` should include the name of the function and the value of its inputs (if it has inputs).
- Place a `console.log(string)` at after each complex calculation so you can see the result is what you think they should be.
Definitions

- The cypher is a bijection (a one-to-one onto map).
- The cardinality (size) of the bijection is 26 (the letters of the English alphabet).
- The letters to be encrypted form a well-ordered set (every non-empty subset of the set has a least element in this ordering).
- An ordinal denotes a position in a sequence ("first", "second", "third", etc.) of a well-ordered set.
- The ordinal of a plaintext letter in the alphabet array is: (A=0, B=1, C=2, ... Z=25).

Quiz: What is the output?

1) a = [];  
2) i = 2;  
3) y = 16;  
4) while (i < 20)  
5) { a.push(i);  
6) text(a, 5, y);  
7) i = i * 3;  
8) y = y + 16  
9) }

(a) (b) (c) (d) (e)
Quiz: What is the output?

1) \( a = [] \);  \( a = [] \rightarrow [1, 1] \rightarrow [1, 1, 5, 5] \)
2) \( i = 1; \)  \( i = 1 \rightarrow 5 \rightarrow 25 \)
3) \( y = 15; \)  \( y = 15 \rightarrow 30 \rightarrow 45 \)
4) while (\( i \leq 20 \))
5)  \{  \( a.push(i); \)  
6)  \}  \( a.push(i); \)
7)  \text{(a, 5, y);}  
8)  \}  \( i = i * 5; \)
9)  \) \( y = y + 15 \)
10) \}

(a)  
(b)  
(c)  
(d)  
(e)  

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,5</td>
<td>1,1,5,5</td>
<td>1,1</td>
<td>1,1,5,5</td>
</tr>
<tr>
<td>1,5,15</td>
<td>1,5,10</td>
<td>1,5,10,15</td>
<td>1,1,5,5,10,10</td>
<td>1,5,10,15,20</td>
</tr>
</tbody>
</table>