Computer Science Correspondence School

discere domi

Lecture 4

CISC 131
Introduction to Programming and Problem Solving
Spring 2020
Caesar's Cipher

Due: **Tuesday, March 31, 2020**
Points: **none**

Solution to Problem from Lecture 3

There are usually several ways to implement the logic in a function. If your implementation of the *html* generator function works correctly, then that is all that is required. Here is how I did it, just to give you another programmer's approach. Make sure you can follow and understand the code.

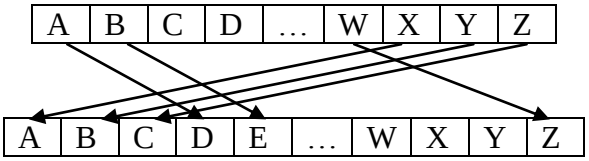
```
function createHTMLElement ( elementType, id, classInfo, content )
{
 if ( elementType === null ) { elementType = ''; }
 if ( id          === null ) { id = ''; }
 if ( classInfo   === null ) { classInfo = ''; }
 if ( content     === null ) { content = ''; }

 elementType = elementType.trim();
 id          = id.trim();
 classInfo   = classInfo.trim();

 if(id.length > 0)        { id        = ' id="' + id + '"'           }
 if(classInfo.length > 0) { classInfo = ' class="' + classInfo + '"' }

 return '<'  + elementType +
        id   + classInfo   +
        '>'  + content     +
        '</' + elementType + '>';

}  // function createHTMLElement
```

Lab Assignment

Caesar's cipher is a very simple encryption technique that can be used with text messages. Ciphers are just one of the many ways that encryption can be done. In the encryption world, the original message is referred to as *plain text* and its coded counterpart is called *cipher text*. The encryption algorithm takes *plain text* as input and produces *cipher text* as output.

Caesar's cipher is not a very good encrypting approach since it does single character substitution. Single character substitution techniques can be easily cracked and the *plain text* discovered. It is, however, a very understandable cipher and pretty easy to program.

The idea is that each character of the *plain text* message is replaced by the character three positions after it in the alphabet. It does this type of mapping:



That is, plain text *A* becomes cipher text *D*, *B* becomes *E* and so on until *X* which becomes *A*, *Y* becomes *B*, and *Z* becomes *C*. The Caesar cipher does everything in uppercase and plain text characters that are not part of the alphabet are not transformed by the encryption algorithm – they are simply copied into the cipher text. For example, if this were the clear text:

```
It's 70 today.
```

then the cipher text would be:

```
LW'V 70 WRGDB.
```

Caesar's cipher is not symmetric in the sense that if you were to take the cipher text produced by the algorithm and use it as plain text input, the cipher would not produce the original plain text input. For example, if the original plain text message was just the letter *A*, the cipher text produced would be the letter *D*. If you then sent *D* as input to the cipher, you would get *G* out, not *A*.

Step 1

Create empty *html* and *Javascript* files. As you have done in the past, test to ensure that there are hooked together correctly. There will be no *css* or additional *html* needed for this assignment.

Step 2

Create a function named *getAlphabet* that returns a *String* containing the uppercase letters of the alphabet in alphabetic order. This function is a convenience and allows you to specify the alphabet only once and use it anywhere just by calling the function.

Step 3

Create a function named *encrypt* that will be passed a *String* containing the plain text. The function will encrypt the parameter using the Caesar cipher and will return a *String* containing the cipher text.

If a character in the plain text is not in the alphabet, that character has no encrypted version and it should be concatenated onto the cipher text result. Inside the loop, you will need both an *if* statement and the use of the modulus operator. The parameter must be converted to upper case before the loop processes it.

Test out your function using the sample data shown above. Test it out by passing the entire alphabet to the function and examine the results. It should be apparent whether the function is working correctly or not.

Step 4

Create a function named *decrypt* that will be passed a *String* containing the cipher text. The function will decrypt the parameter using the Caesar cipher and will return a *String* containing the plain text.

If a character in the cipher text is not in the alphabet, that character has no decrypted version and it should be concatenated onto the plain text result. Inside the loop, you will need both an *if* statement and the use of the modulus operator. The parameter must be converted to upper case before the loop processes it. Test this thoroughly. Take some text, encrypt it, and then decrypt. The decrypted version and the original text must be the same other than that the decrypted version will be in uppercase.

Step 5

Notice how similar the *encrypt* and *decrypt* functions are. Study them and then write a function named *encryptDecrypt*. The function is passed a *String* and an integer. Depending on the value

of the integer, the function will either encrypt the *String* parameter or decrypt it. Hint, the value of the integer actual parameter should always be greater than zero. The function is very similar to the *encrypt* and *decrypt* functions and contains only a single loop. Test this thoroughly. I should produce results that are the same as the separate encrypt and decrypt functions.

What You Must Hand In
Nothing