WAD Lab Test 1 (1.5 hours) [20 marks]

General Instructions:

- You can refer to any offline resources already on your laptop, but you must disable all networking and Bluetooth connections during the test. You must not communicate with anyone via any means during the test.
- Just before the test, you will be given instructions by the invigilator as to how to obtain resource files required for the lab test and how to submit your solutions.
- No questions will be entertained during the test. If necessary, make your own assumptions.
- You are allowed to use only standard PHP classes and functions in your solutions – do not use any third-party libraries.
- Use meaningful names for classes, methods, functions and variables, as well as indent your code correctly. Use 4 spaces for indentation. Otherwise, you may attract penalty of up to **20%** of your score for the corresponding question.
- You **MUST** include your name as author in the comments of all your submitted source files. Failure to do so WILL attract a penalty of up to **20%** of your score for the corresponding question.

  For example, if your registered name is "TAN So Tong" and email ID is tan.sotong.2017, include the following comment at the beginning of each source file you write.

  ```
  <!--
        Name:  TAN So Tong
        Email: tan.sotong.2017
  -->
  ```
- You may wish to comment out the parts in your code which cause errors. But commented code will not be marked.

DO NOT TURN OVER UNTIL YOU ARE TOLD TO DO SO

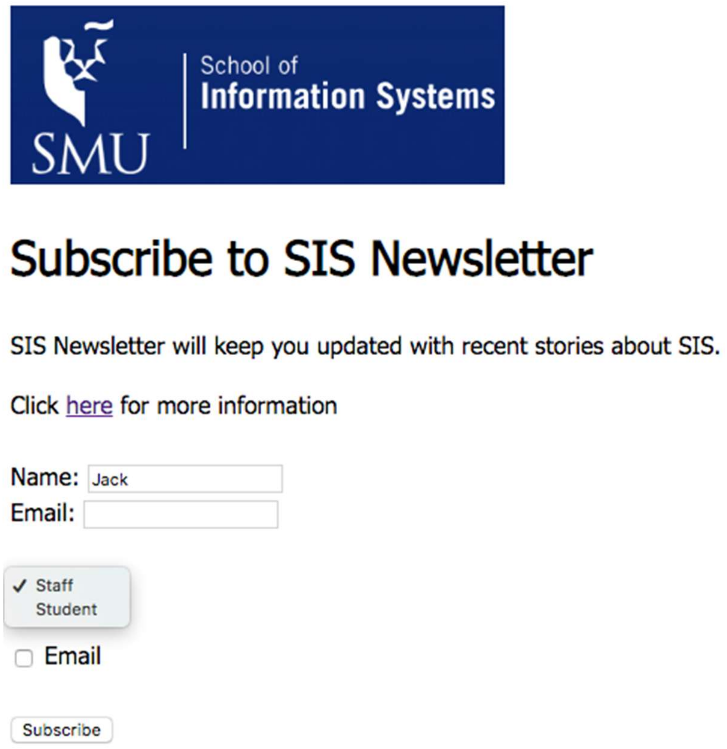
Question 1 [Difficulty Level: *] [8 marks]

Given:

```
1. q1.html (Edit this file. We will mark this file)
2. q1.php  (Edit this file. We will mark this file)
3. logo.png
```

Task 1 [6 marks]

Edit **`q1.html`** such that it displays the web page as shown below:



Subscribe to SIS Newsletter

SIS Newsletter will keep you updated with recent stories about SIS.

Click here for more information

Name: Jack
Email:

✓ Staff
Student

Email

Specifically, it shows:

a. SMU logo image. Use the given image **`logo.png`** in the resource folder.

b. the header "Subscribe to SIS Newsletter". Use **HTML Header 1** tag.

c. the paragraph "SIS Newsletter will keep you updated with recent stories about SIS."

d. a hyper-link "Click here for more information". Let it link to **`q1.html`** itself.

e. has a form that shows:

   i. two input fields with type='text'

   ii. a drop-down list with two options: 'Staff' and 'Student'

   iii. a checkbox labeled with 'Email'

   iv. a SUBMIT button 'Subscribe'; upon clicking this button, the form should submit to **`q1.php`** via HTTP POST.

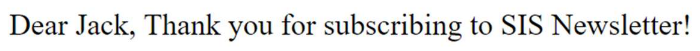
Task 2 [2 marks]

Edit **`q1.php`** such that it displays the following message when the form in **`q1.html`** has been submitted:

Dear <name>, Thank you for subscribing to SIS Newsletter!

where <name> is the placeholder for the value entered in the "name" input field in **`q1.html`**

A sample output of **`q1.php`** is shown below:



```
Dear Jack, Thank you for subscribing to SIS Newsletter!
```

Question 2 [8 marks]

Given:

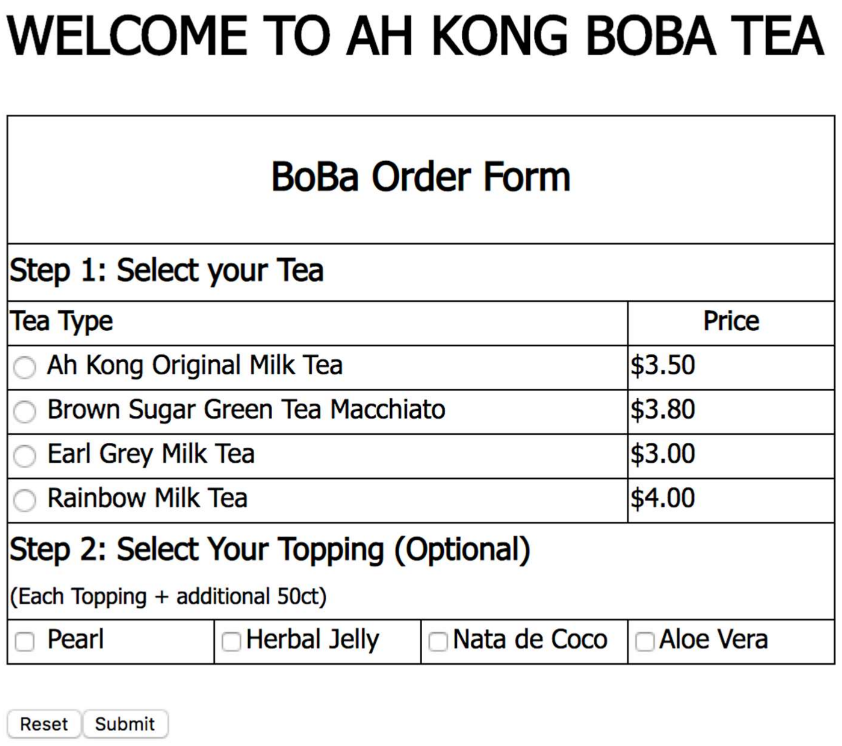
- `q2.html     (Edit this file. We will mark this file)`
- `q2.php      (Edit this file. We will mark this file)`

Task 1 [4 marks, Difficulty Level: */]**

Edit **`q2.html`** to have a form that:

a. contains radio buttons for the selection of **Tea Type**.
b. contains checkboxes for the selection of **Toppings**.
   - Toppings are optional (the user can choose NO toppings; **or**, the user can choose 1 or more toppings)
   - Each topping is at $0.50 each.
c. Clicking on the text of the radio button/check boxes will select the radio button/check boxes.
d. submits to **`q2.php`** via HTTP POST, upon clicking the SUBMIT button.

Your **`q2.html`** output should look like below:



WELCOME TO AH KONG BOBA TEA

BoBa Order Form			
Step 1: Select your Tea			
Tea Type			Price
○ Ah Kong Original Milk Tea			$3.50
○ Brown Sugar Green Tea Macchiato			$3.80
○ Earl Grey Milk Tea			$3.00
○ Rainbow Milk Tea			$4.00
Step 2: Select Your Topping (Optional) (Each Topping + additional 50ct)			
☐ Pearl	☐ Herbal Jelly	☐ Nata de Coco	☐ Aloe Vera

Task 2 [4 marks, Difficulty Level: **]

Edit **`q2.php`** so that it:

a. displays the following error message if tea type has not been selected:

   Please select a tea type!

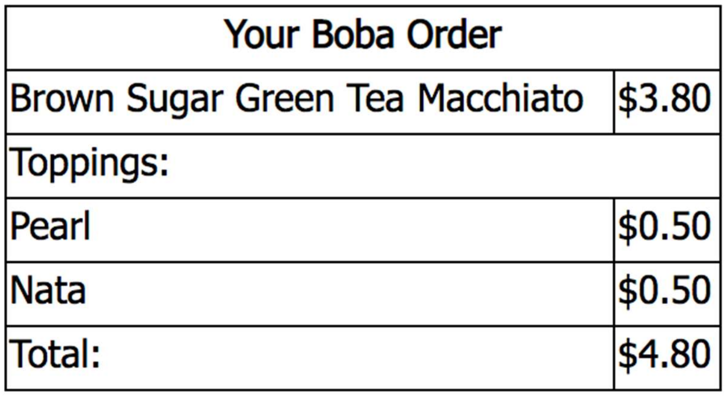
b. calculates the cost of the Tea order if tea type and topping choices (optional) have been selected.

c. prints the Receipt in an HTML table that looks like the diagram below:

   - displays total price in 2 decimal places. See PHP function `number_format()`

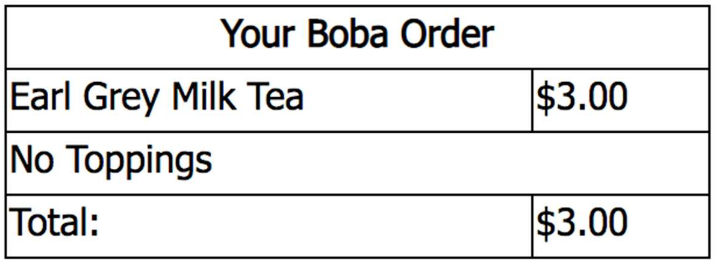
The sample runs of **`q2.html`** and **`q2.php`** are shown below:

`q2.php` – with toppings selected in **`q2.html`**



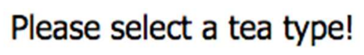
Your Boba Order	
Brown Sugar Green Tea Macchiato	$3.80
Toppings:	
Pearl	$0.50
Nata	$0.50
Total:	$4.80

`q2.php` – with no topping selected in **`q2.html`**



Your Boba Order	
Earl Grey Milk Tea	$3.00
No Toppings	
Total:	$3.00

`q2.php` – with no Tea type selected in **`q2.html`**



Please select a tea type!

Question 3 [Difficulty Level *]** **[4 marks]**

Given:

1. `q3-encrypt.html`
2. `q3-decrypt.html`
3. `q3.php     (Edit this file. We will` **`ONLY`** `mark this file)`

In cryptography, encryption is a function that converts a given **message**, using a **key**, into a secret code known as **cipher**. Decryption is a function that converts the cipher back to the original message, using **the same key**. For example,

```
Encrypt("secret message", "key")           ->  "EgcAEwQVQQwEEhIABgQ="
Decrypt("EgcAEwQVQQwEEhIABgQ=", "key")      ->  "secret message"
```

In PHP, **the XOR operator ^** can be used to implement such an encryption and decryption function. For example,

```
$cipher  = $message ^ $key; // encrypt
$message = $cipher ^ $key;  // decrypt
```

where $cipher, $message, and $key are strings.

It must be ensured that the key has the same length as the message. For example, if the key provided by the user is "`abc`" and the message is "`secret`", one possible key would be "`abc***`", where the original key is padded with three '`*`' characters.

<u>**Instructions**</u>

NOTE: We will be using a script to automatically test your q3.php file. Therefore, please make sure that your encrypt function returns an array of encrypted chunks and your decrypt function returns the decrypted message. Refer to the instructions and examples given below.

Part 1

`q3-encrypt.html` provides a form in which the user can provide a **message** and the **key**, and it contains an "Encrypt" button, which submits to **`q3.php`**.

Assume the following:

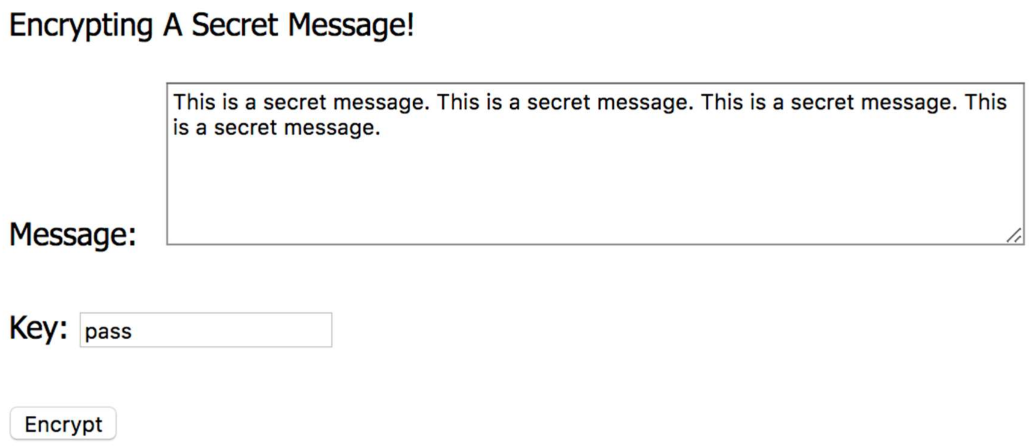
- the message contains a **minimum of 2** sentences
- each sentence contains **no more than 100 characters**
- the message always ends with a full stop; a sentence always ends with a full stop. (see examples below)
- if there is any leading space(s) in a sentence, it is to be considered as part of the sentence (e.g., " There is a leading space in this sentence.")
- the key's length is always shorter than the length of any sentence of the message

Edit **`q3.php`** such that, when the user clicks "Encrypt" button in **`q3-encrypt.html`**, it encrypts the message, **chunk-by-chunk**, using the key provided by the user. That is, it splits a given message into chunks and encrypts them one-by-one. It then displays the encrypted chunks.

A. You need to implement a function `encrypt_a_chunk($chunk,$key)`:
   i. Pad the key so that it has the same length as the chunk
   ii. Encrypt the chunk using PHP XOR operator ^
   iii. The encrypted chunk from the previous step is not human-readable. Hence, use PHP function `base64_encode()` to make it human-readable, e.g., `base64_encode($cipher)`
   iv. Return the human-readable encrypted chunk, from step (iii)

B. You need to implement a function `encrypt($message,$key)` that breaks the message into chunks, encrypts each chunk by calling `encrypt_a_chunk($chunk,$key)`, and returns the encrypted chunks as an array. **It is required that:**
   i. maximum size of a chunk is 100 characters
   ii. each chunk may contain one or more sentences but they must be complete sentences (note: a complete sentence is always less than or equal to the maximum chunk size. See examples below).

Example 1: Encrypting a message that has two chunks



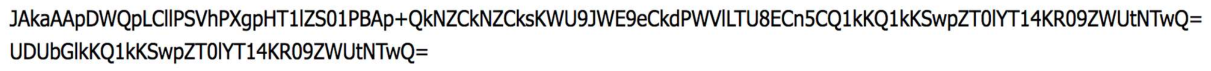
Encrypting A Secret Message!

Message: This is a secret message. This is a secret message. This is a secret message. This is a secret message.

Key: pass

`q3-encrypt.html`

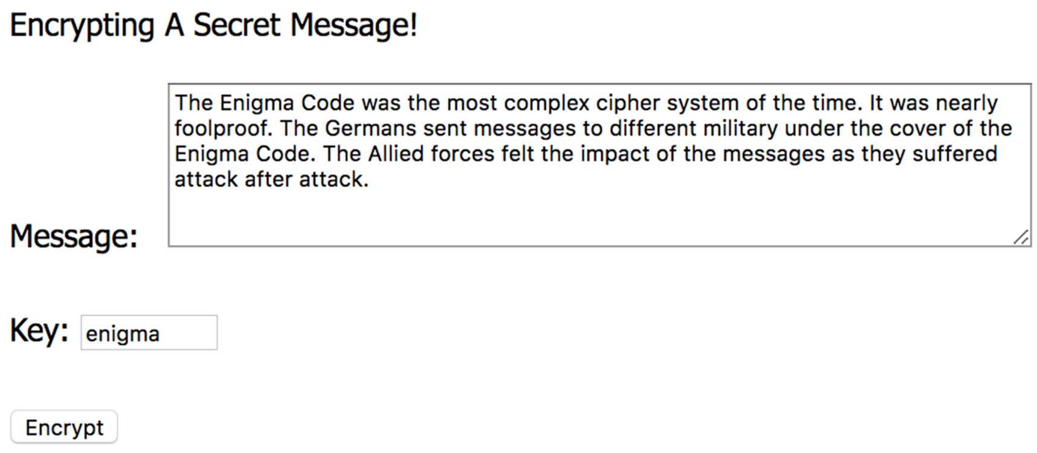
On clicking the "Encrypt" button, it should result in:



```
JAkaAApDWQpLClIPSVhPXgpHT1lZS01PBAp+QkNZCkNZCksKWU9JWE9eCkdPWVlLTU8ECn5CQ1kKQ1kKSwpZT0lYT14KR09ZWUtNTwQ=
UDUbGlkKQ1kKSwpZT0lYT14KR09ZWUtNTwQ=
```

`q3.php`

Example 2: Encrypting a message that has three chunks



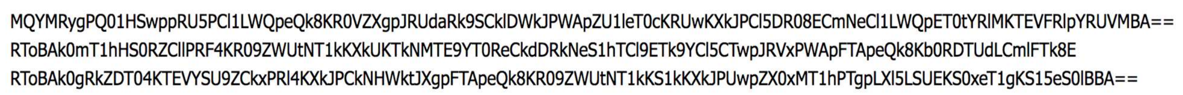
Encrypting A Secret Message!

Message: The Enigma Code was the most complex cipher system of the time. It was nearly foolproof. The Germans sent messages to different military under the cover of the Enigma Code. The Allied forces felt the impact of the messages as they suffered attack after attack.

Key: enigma

`q3-encrypt.html`

On clicking the “Encrypt” button, it should result in:



```
MQYMRygPQ01HSwppRU5PCl1LWQpeQk8KR0VZXgpJRUdaRk9SCklDWkJPWApZU1leT0cKRUwKXkJPCl5DR08ECmNeCl1LWQpET0tYRlMKTEVFRlpYRUVMBA==
RToBAk0mT1hHS0RZCllPRF4KR09ZWUtNT1kKXkUKTkNMTE9YT0ReCkdDRkNeS1hTCl9ETk9YCl5CTwpJRVxPWApFTApeQk8Kb0RDTUdLCmlFTk8E
RToBAk0gRkZDT04KTEVYSU9ZCkxPRl4KXkJPCkNHWktJXgpFTApeQk8KR09ZWUtNT1kKS1kKXkJPUwpZX0xMT1hPTgpLXl5LSUEKS0xeT1gKS15eS0lBBA==
```

q3.php

Note: The sample messages above are provided as comments in **`q3-encrypt.html`**

Part 2

`q3-decrypt.html` provides a form in which the user can provide the encrypted chunks and the key; and it contains an “Decrypt” button, which submits to `q3.php`.

Assume the following:

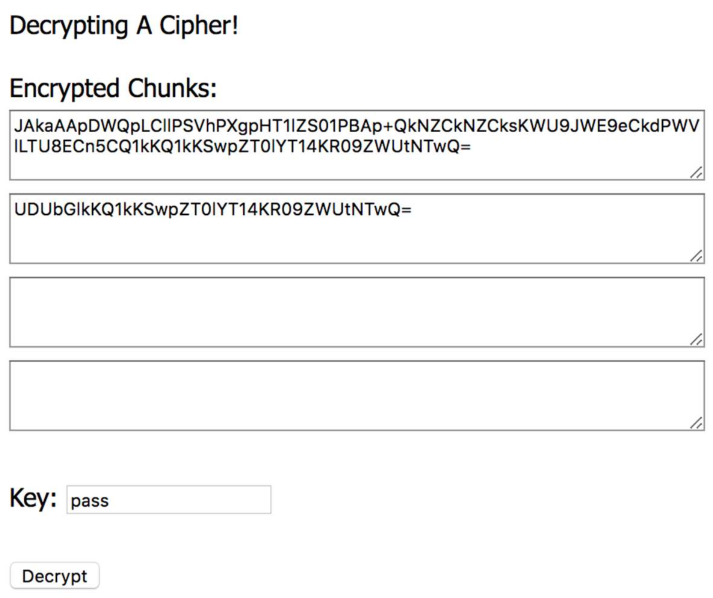
- User provides the encrypted chunks observed in Part (1) above
- There are no more than four encrypted chunks

Edit `q3.php` such that when the user clicks “Decrypt” button in `q3-decrypt.html`, it decrypts the encrypted chunks using the key provided by the user; and displays the original message.

A. You need to implement a function `decrypt_a_chunk($encrypted_chunk, $key)` where `$encrypted_chunk` is the result from Part (1) above. Hint: use `base64_decode()` and PHP XOR operator ^

B. You need to implement a function `decrypt($encrypted_chunks,$key)` that calls `decrypt_a_chunk($encrypted_chunk,$key)` for every element of array `$encrypted_chunks`, **concatenates** the return values, and returns the eventual string as the decrypted message (see examples in the following).

Example 1: Decrypting two encrypted chunks



Decrypting A Cipher!

Encrypted Chunks:

```
JAkaAApDWQpLClIPSVhPXgpHT1lZS01PBAp+QkNZCkNZCksKWU9JWE9eCkdPWVlLTU8ECn5CQ1kKQ1kKSwpZT0lYT14KR09ZWUtNTwQ=
```

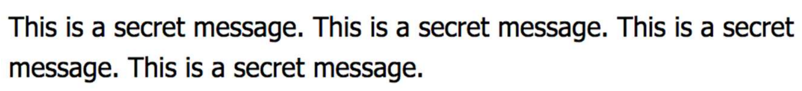
```
UDUbGlkKQ1kKSwpZT0lYT14KR09ZWUtNTwQ=
```

Key: pass

Decrypt

q3-decrypt.html

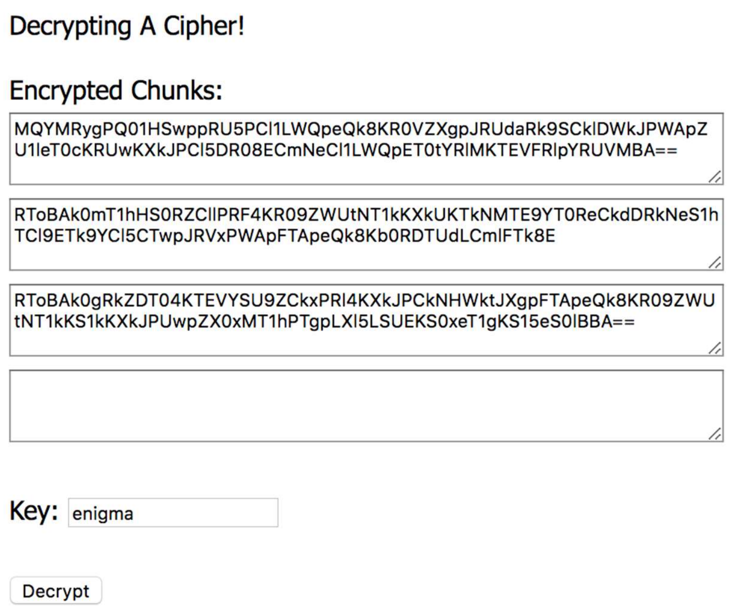
On clicking the “Decrypt” button, it should result in:



This is a secret message. This is a secret message. This is a secret message. This is a secret message.

q3.php

Example 2: Decrypting three encrypted chunks



Decrypting A Cipher!

Encrypted Chunks:

```
MQYMRygPQ01HSwppRU5PCl1LWQpeQk8KR0VZXgpJRUdaRk9SCklDWkJPWApZU1leT0cKRUwKXkJPCl5DR08ECmNeCl1LWQpET0tYRlMKTEVFRlpYRUVMBA==
```

```
RToBAk0mT1hHS0RZClIPRF4KR09ZWUtNT1kKXkUKTkNMTE9YT0ReCkdDRkNeS1hTCl9ETk9YCl5CTwpJRVxPWApFTApeQk8Kb0RDTUdLCmlFTk8E
```

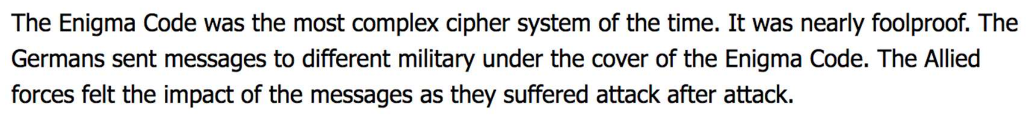
```
RToBAk0gRkZDT04KTEVYSU9ZCkxPRl4KXkJPCkNHWktJXgpFTApeQk8KR09ZWUtNT1kKS1kKXkJPUwpZX0xMT1hPTgpLXl5LSUEKS0xeT1gKS15eS0lBBA==
```

Key: enigma

Decrypt

q3-decrypt.html

On clicking the "Decrypt" button, it should result in:



The Enigma Code was the most complex cipher system of the time. It was nearly foolproof. The Germans sent messages to different military under the cover of the Enigma Code. The Allied forces felt the impact of the messages as they suffered attack after attack.

`q3.php`

Note: The sample encrypted chunks above are provided as comments in `q3-decrypt.html`

- END -