1. Pick a password. Remove all not alphabetical characters and remove duplicate letters so that at most only one occurrence of each letter of the alphabet appears. For example "**Question Authority**" becomes "**QUESTIONAHRY**" (case doesn't matter).
2. Create a 5 by 5 matrix and enter each letter of the password in the spaces. There are multiple ways of placing the password, but the most common way to do it is enter the letters starting at the top, left-most position and move right across the first row, then second row, etc. Here is my matrix using my previous example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q | U | E | S | T |
| I | O | N | A | H |
| R | Y | - | - | - |
| - | - | - | - | - |
| - | - | - | - | - |

1. Using your wonderful analytical mind, you can quickly realize that there a 25 total spots in this matrix. Well what are we going to do with the rest of the spots? Fill them with the rest of the alphabet, of course! But wait... there's 26 letters of the alphabet? Who's gonna get the shaft? Well that's a question of preference. Some people using Playfair replace all instances of **J** with **I**, while others completely discard instances of the letter **Q**. Well, I'll give you both examples. For the upcoming instructions, I will be using the **replace J with I** scheme.

**Replace J with I**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q | U | E | S | T |
| I | O | N | A | H |
| R | Y | B | C | D |
| F | G | K | L | M |
| P | V | W | X | Z |

**Omit Q**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| U | E | S | T | I |
| O | N | A | H | R |
| Y | B | C | D | F |
| G | J | K | L | M |
| P | V | W | X | Z |

Now that we have the password matrix, we can encript text.

1. Great, so we have this pretty square matrix. This will be what we use to encrypt text given to us. So find a string we want to encrypt... how about "**Joel enjoys programming on Friday nights! Yah!**". To get ready, we need to split the text we are encrypting into two-letter groups (called **digraphs**). As before, pull out all non-alphabetical characters (multiple instances of the same letter is okay though!). Here is what my example would look like: "**IO EL EN IO YS PR OG RA MX MI NG ON FR ID AY NI GH TS YA HX**". There are three edge-cases you need to consider.
   1. If you run into a digraph that has two of the same letters, split the double letters up with an uncommon character. **X** is most commonly used.So for my example, I ran into the digraph **MM** (from "progra**mm**ing"). So I inserted an **X** into my text phrase, as you can see.
   2. The other edge case you need to consider is if there are an odd number of letter in the text phrase. When this happens, just tack an **X** on the end of the phrase. In my running example, I was stuck with the letter **H**(from "Ya**h"**) without a partner. So, I gave **Mr. H** a **Ms. X**, as any moral programmer would.
   3. Since I used the **replace J with I** rule to create my 5 by 5 matrix, I need to apply this same rule to my input text. As you can see, all the **J's**in my input text (from "**J**oel" and from "en**j**oys") got zapped into **I's**. Yes, my name is now **Ioel**. Wonderful. Keep in my that if I was using the **omit Q** rule, any **Q's** in my input text would be discarded.
2. So now we have a list of digraphs which roughly resemble our phrase which we plan on encrypting. So now comes the fun part! Let's encrypt these babies! Every digraph will be encrypted into another corresponding digraph. There are three cases we need to look at. To illustrate them, I will use my running example.
   1. I'll take the first digraph in my list (in my case it is **IO**) and find **I**and **O** on my grid. The two letters appear on the **same row** of the matrix.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q | U | E | S | T |
| **I** | **O** | N | A | H |
| R | Y | B | C | D |
| F | G | K | L | M |
| P | V | W | X | Z |

* 1. To get the corresponding encrypted digraph when the two letters are on the same row, **shift one space to the right** from each letter. So **I** would become**O**and **O**would become **N**. If the rightward shift takes you off the edge of the matrix, loop around to the other side. For example, the digraph **IH** would become **OI**. **KM** would become **LF**.
  2. The next case you need to look at is if the two letters of a digraph are in the **same column**. Let's take the third digraph in my example, which is **EN**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q | U | **E** | S | T |
| I | O | **N** | A | H |
| R | Y | B | C | D |
| F | G | K | L | M |
| P | V | W | X | Z |

* 1. Very similar to the **same row** case, in this case you **shift one space down**. Again, if the shift takes you off the edge of the matrix, just loop back around to the top. **EN** would become **NB**. **CL** would become**LX**. **HZ** would become **DT**.
  2. The last case you need to look at is if the two letters are not in the same row or column. In this case, we say they form the opposite corners of a rectangle. To get the digraph, you just take the other two corners of the rectangle. Confused? Lemme give you an example. Let's take the digraph **RA**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q | U | E | S | T |
| I | O | N | **A** | H |
| **R** | Y | B | C | D |
| F | G | K | L | M |
| P | V | W | X | Z |

* 1. If you get imaginative for a moment, picture the **R** and **A** spaces in the matrix above forming a rectangle, where **A** is the top-right corner and **R** is the bottom-left. See it? Good! To get the encrypted digraph for this example, just take the letter in the corners to the right or left of **R** and **A**. So in this case, **A** becomes **I** and **R** becomes **C**. Notice that **R does not become I**. You take the corner that is to the right or left of the letter you are currently encrypting. Here are some more examples to make sure you get it. The digraph **QK** becomes **EF**. **HL** becomes **AM**.

1. Once you've done that to all of your digraphs, you can slam all of the encrypted digraphs together. That's your encrypted phrase. For my example, "**Joel enjoys programming on Friday nights! Yah!**" becomes "**ONSKNBONCUQFYVCILZFHOKNAPFHROCAOMOQTCOAZ**". But wait, that's complete gibberish! That's the point, stupid head! Unless you have the password matrix, it is pretty hard to decrypt the text.

Well how do you decrypt, anyways? Well, decryption is very similar to the encryption process. Just follow the following notes, and you'll be in business.

* Given you have the password, create the 5 by 5 password matrix just as you did above. You need to make sure to use the same omission rule (i.e. **omit Q**or **replace J with I**) as you used to encrypt the text.
* Separate the encrypted phrase into digraphs and use the **inverse of the cases** **I outlined above** to decrypt each encrypted digraph.
  + For digraphs which occur in one row, **shift one space to the left**(looping around if necessary).
  + For digraphs which occur in one column, **shift one space up** (looping if necessary).
  + For digraphs which form a rectangle, use the same exact process as in the encryption process (use the opposite corner in the same row).