INTRODUCTION TO COMPUTER SCIENCE I
LAB 6
Caesar Cipher

For this lab, you'll be introduced to working with the `char` data type, all while practice your use of arrays. As an excuse to work with arrays of characters, we'll do some basic *cryptography*.

1 Cryptography

Cryptography is, loosely speaking, the study of methods for taking "human readable" data (*cleartext* or *plaintext*) and then scrambling (*encrypting*) it into an obfuscated form (*ciphertext*) that cannot easily be read. Of course, such obfuscation is useless unless there is some means by which the ciphertext can be restored (*decrypted*) to its original cleartext form.

A *cipher* defines a *one-to-one correspondence* between plaintexts to ciphertexts. That is, it defines a pair of algorithms, one for encryption and one for decryption, that respectively scramble and unscramble data. Each cipher uses a single parameter called the *key* that is used as part of the encryption and decryption calculations. For a given ciphertext, the same key must be used to decrypt it as was used when creating it during encryption.[1]

The ciphers that we are going to use are *monoalphabetic substitution ciphers*. Specifically, for these ciphers, encryption is accomplished by replacing each character in the plaintext with a different letter in the ciphertext. Therefore, the lengths of the plaintext and the ciphertext are equal, and both encryption and decryption can each be performed one character at a time.

2 The Caesar Cipher

For this first part of this project, we will use the *Caesar cipher* (also known as a *shift cipher*). To encrypt a message using this cipher, each character of the cleartext is *shifted* forward by k characters in the alphabet, thus producing a ciphertext. Likewise, decrypting a message requires shifting each character in the ciphertext backwards by the same k characters in the alphabet. Moreover, since the alphabet of characters is a fixed-sized list, it is possible to shift "off the end" of the list. Whenever this situation arises, the *shift* operation "wraps around" to the beginning of the alphabet.

To see how this cipher works, consider only the 26 uppercase letters of the English alphabet. Furthermore, consider using the key value $k = 5$. The last 5 characters in this alphabet, when shifted, are wrapped-around to the first five characters. So, the correspondence between the cleartext and ciphertext characters would be:

Cleartext char	A	B	C	D	E	F	G	…	S	T	U	V	W	X	Y	Z
Ciphertext char	F	G	H	I	J	K	L	…	X	Y	Z	A	B	C	D	E

Thus, the following message, if encrypted using the same key of $k = 5$ would be:

[1]The use of the same key value for encryption and decryption makes a cipher *symmetric*. There are *asymmetric* ciphers, most notably public-key cryptography, where different keys are used for encryption and decryption, but those are beyond the scope of this course. Consider the Networks and Cryptography course.

Cleartext message	`THEQUICKBROWNFOX`
Ciphertext message	`YMJVZNHPGWTBSKTC`

Implementation hints: In order to write a program that performs this type of encryption and decryption, there are two valuable observations of which you need to be aware:

1. **A `char` is an integer in disguise:** Since each character is really a numeric value, then you can perform all of the common arithmetic operations on them. You can add k to shift forward for encryption, and you can subtract k to reverse the shift for decryption.[2]

2. **Our alphabet is 95 characters:** Each `char` datum is a numeric value between 0 and 255 (inclusive), but the *printable characters*—those that you can see when printed to the screen—make up a subset of only 95 of the 256 possible character values. Specifically, our alphabet comprises character values 32 through 127. So, when you add k, you must be sure that the resulting sum is made to "wrap around" to stay within this range.

3 Getting started

Begin like so:

1. **Login** to `remus` or `romulus`.

2. **Open a terminal** to get a shell prompt.

3. **Make a directory** named `lab-6` (using `mkdir`) and change into it (using `cd`).

4. **Copy** source code from my directory:
```
$ cp ~sfkaplan/public/COSC-111/lab-6/CaesarCipher.java .
```

5. **Open and edit** this source code file with `emacs`.

4 Your assignment

4.1 What you must write

The file `CaesarCipher.java` already contains a significant portion of the needed source code. The one missing piece is the body of the method `cipher()`. This method is passed an array of characters (via the parameter `text`), as well as the desired shift to be performed on each character (via the parameter `key`). This method must…

- Create a new array that will hold the encrypted/decrypted copy of the `text` passed.

- Traverse the `text` array, shifting each character by the `key` and storing the result into the same position in the new array.

- Return the new, encrypted/decrypted array.

[2]For the program you will create in this lab, it will be up to the user to provide $-k$ in order to decrypt a message that was encrypted using k.

4.2 How to test your code

When you compile and run your `CaesarCipher` program, its use looks like this…

```
$ java CaesarCipher 5
The quick brown fox<ctrl-D><ctrl-D>
---Begin output---
Ymj%vznhp%gwt|s%kt}
---End output---

$ java CaesarCipher -5
Ymj%vznhp%gwt|s%kt}<ctrl-D><ctrl-D>
---Begin output---
The quick brown fox
---End output---
```

Notice a few important things:

- On the command-line, when you run `CaesarCipher`, you must include the key value at the end before pressing *enter* to start the program. Feel free to look at `main()` within the program to see how entering the key value this way works.
- After typing some text, you must press the key combination *control-D* **twice** in order to end your input.
- To decrypt, you enter (or copy-and-paste) the encrypted text while providing the negated key value (that is, $-k$).

5 How to submit your work

Use the CS submission systems to submit your work. Specifically, you will need to submit your `CaesarCipher.java` file. You may use either of the following two methods, while connected to `remus` or `romulus`, to use the submission system:

- **Web-based:** Visit the submission system web page.
- **Command-line based:** Use the `~lamcgeoch/submit` command at your shell prompt.

This assignment is due on Tuesday, Apr-12, 11:59 pm.