CSC 116
Fundamentals of Programming with Engineering Applications II
Assignment

Note 1 **This assignment is to be done individually**

Note 2 You can discuss the assignment with others. However sharing and copying code is prohibited (this includes looking at somebody else's code).

A note on Academic Integrity and Plagiarim

Please review the following documents:

- Standards for Professional Behaviour, Faculty of Engineering:
  `https://www.uvic.ca/engineering/assets/docs/professional-behaviour.pdf`
- Policies Academic Integrity, UVic:
  `https://www.uvic.ca/students/academics/academic-integrity/`
- Uvic's Calendar section on Plagirism:
  `https://www.uvic.ca/calendar/undergrad/index.php#/policy/Sk_0xsM_V`

Note specifically:

> Plagiarism
> Single or multiple instances of inadequate attribution of sources should result in a failing grade for the work. A largely or fully plagiarized piece of work should result in a grade of F for the course.

A program you submit will be considered a **piece of work**. You are responsible for your own submission, but you could also be responsible if somebody plagiarizes your submission.

Objectives

After completing this assignment, you will have experience with:

- Basic input and output
- Manipulation of strings

Your task, should you choose to accept it

One-time-pad is a method to encrypt information that is proven unbreakable (as long as it is used properly). Go to Wikipedia and learn a bit about it: `https://en.wikipedia.org/wiki/One-time_pad`.

Your job is to write a program that implements one-time-pad encryption.

One-time-pad

The algorithm is very simple: Given two strings, the *key* and the *message* such as $length(key) >= length(message)$, it outputs a string *encrypted*. The string *encrypted* should satisfy the following precondition:

- The key should be at least as long as the message to encrypt:

$$\text{length}(\texttt{key}) \geq \text{length}(\texttt{message})$$

And have the following postconditions:

- The encrypted message should have the same length as the original message:

$$\text{length}(\texttt{encrypted}) == \text{length}(\texttt{message})$$

- Every character in the encrypted message is the result of XORing the corresponding characters in the message and the key:

$$\texttt{encrypted}[i] == \texttt{message}[i] \text{ XOR } \texttt{key}[i]$$

- One-time-pad is a symmetric algorithm: the decrypted message is the result of encrypting the encrypted message with the same key:

$$\texttt{decrypted}[i] == \texttt{encrypted}[i] \text{ XOR } \texttt{key}[i]$$

- This results in the following invariant:

$$\texttt{message} == \text{one\_time\_pad}(\text{one\_time\_pad}(\texttt{message}, \texttt{key}), \texttt{key})$$

Your code

Download the provided code and tests from Connex under resources. Your task is to complete the program provided, including:

1. Writing a function to do the encoding/decoding which takes two parameters (the message and the key), and returns the encrypted message.

The complete program should comply with the following specification.

Input

The input to your program is:

1. The first line in the input is the key. This key will be used to encrypt all the messages in the input.
2. Every remaining line in the input is a message that should be encrypted. There might be zero or more lines to encrypt. Some lines might contain the empty string.

Output

The output should satisfy the following constraints (see examples below and test cases for more information):

1. The first line of the output should be the input key
2. For every message to encrypt: it should output, in one line each:
   - The message.
   - The encrypted message. Because this message might contain characters that are not printable in the console, if the a character value is less than 32 or bigger than 126, it should print its numerical value surrounded by parenthesis; otherwise the character should be printed as usual (see below).
   - The result of decrypting the encrypted message (this string should be identical to the original message). Your program should output the result of decrypting the encrypted message.
3. The program's exit status (that is, the return value of `main`) should be 0 in the case of success.
4. If, at any point, there is a message with $length(message) > length(key)$ the program should terminate with an error message. In this case the exit status of the program should be 1. You can use `exit(1);` for this purpose. Alternatively the return value of `main` can be 1.

Example run 1:
Given the following input:

```
ab0123456789
XMCKL
hello
```

Your program should output:

```
Original key [ab0123456789]
Original message [XMCKL]
Encrypted [9/sz~]
Decrypted [XMCKL]
Original message [hello]
Encrypted [(9)(7)\]]]
Decrypted [hello]
```

Example run 2:
Given the following input:

```
000
XMC
hello
A
```

Your program should terminate with exit status 1 (see above) and output:

```
Original key [000]
Original message [XMC]
Encrypted [h}s]
Decrypted [XMC]
Original message [hello]
Error: the length of the key [3] is smaller than the length of the message to encrypt [5]
```

In `linux.csc.uvic.ca` you can test the error code of your program by using `echo $?` immediately after you run your program:

```
echo $?
```

Tests

The provided starter files include a set of tests (in the `tests` directory). You should use these tests to verify your code.

The *.input are the input files to your program. The *.expected are the expected output. Note that there are two sets of expected output files. **Use the one that corresponds to your architecture**. MacOS and Linux users should use the unix set. For each test, the output of your program **should be identical** to the one provided. The automatic testing will use the appropriate set of expected files.

When we say that the output should be identical, we mean that:

- the byte streams of both files should be identical, and
- files should have exactly the same size.

If they do not, you run the risk of losing marks (potentially all marks).

Hints

1. In C++, the XOR operation is `^`
2. Convert a `char` to `std::string` using `std::to_string()`.
3. Use `std::getline` to read one line from standard input.
4. Remember, your function `main` should always return 0 in case of success.
5. Learn to use diff tool to compare your output to the expected output (google for 'diff tool windows', for example). Unixes (including MacOS have a tool call `diff`.)

What compiler to use

Your program will be graded by running in the "console" in the computer `linux.csc.uvic.ca` (as described in the first lab). You can develop your code in your computer, but you should always verify that it runs on `linux.csc.uvic.ca`. You should have learned to log-in to `linux.csc.uvic.ca` in the first lab.

Your program will be compiled with `clang++` using the following command line options:

```
-std=c++17 -Wall --pedantic -Werror
```

Note that the `-Werror` option will instruct the compiler to turn warnings into errors (i.e. your program will not compile if there are compilation warnings). If your program does not compile you will receive no marks.

Evaluation

Solutions should:

1. Be Correct. They should pass all the tests we have provided.
2. Be in good style, including indentation and line breaks
3. not use `[]` to access each character in the string. If you do, you will lose marks. Use `.at()` instead.

What to submit

- Using conneX, submit your version of the file `onetimepad.cpp`. **Do not submit any other files**.
- You should assume that your submission was received correctly only if both of the following two conditions hold.
  - You receive a confirmation email from conneX.
  - You can download and view your submission, and it contains the correct data.
- In the event that your submission is not received by the marker, you will need to demonstrate that both of the above conditions were met if you want to argue that there was a submission error.

Re-submission

If you submitted a solution by the original deadline, and you decide that you can do better, you have the opportunity to resubmit your solution again. Your final grade will be the average of the two submissions. We will release details on how to do this when you receive the grade for this assignment.