This document is “living”. As we figure out what questions are frequently-asked, we’ll slowly add to this document.

### Task: Caesar Cipher Encoder

#### I can’t run the project! It says “utils.h” file not found.

We’ll re-emphasize this bit from the guide:

**Use the make commands to run and test your project**, not the run button or typing the clang++ command manually. We’ll tell you the relevant make commands at various places in the project guide when you’re ready to use them.

#### My strings are turning into a bunch of numbers at some point, but I don’t use any numbers?

One possible cause is due to an oddity of C++’s cctype functions. They work on int, not char!

If we directly output the result of e.g. toupper into a stringstream, then the stringstream actually takes in the string representation of the number. For example, ss << toupper('a') doesn’t put “A” into the stream; it puts “65” (the [ASCII value](https://www.asciitable.com/) of “A”). We can resolve this by *casting*: ss << (char) toupper('a').

#### My CaesarEnc\_FullCommand\_Main tests run forever?

This may be due to mixing ways of reading input from cin. As the project guide says, you should always usegetline to read from cin, rather than >>. If you use >>, this leaves the ending space or newline in the stream!

### Task: Caesar Cipher Decoder

#### My output seems correct, but the test is telling me that it’s wrong?

There’s a few different reasons that we’ve seen for this, but usually they come out to one of…  
:

* Not including a space or newline after the colon in decrypt:.
* Not using putTogetherWithSpaces (you wrote the helper function – time to use it!)
* Not printing each possible decrypted text on its own line

### Task: Representing a Substitution Cipher, Encoding

(No FAQ for this part yet!)

### Task: Substitution Cipher Decoder

#### I can’t seem to get the function to work consistently. It’s not really close to English or works inconsistently?

This has more possible causes than the other questions, but one reason could be due to your usage of scoreString. Specifically, it assumes that the input “has only uppercase letters”. For the text that you’re decrypting, this may not be the case! Often, it will have spaces and punctuation as well.

Another possible reason is that you’ve misinterpreted the line in the algorithm “While fewer than 1500 trials in a row have not resulted in a replacement…”. This doesn’t mean “1500 trials total”. Another way of phrasing this would be “Keep track of the number of times you haven’t replaced the key. When this exceeds 1500, stop.”

#### My substitution cipher decryption runs too quickly and gives a bad result?

There’s a few possible causes, but if you look at the Englishness scores in the guide, you’ll notice that they're all negative. If you start with a “best score” of 0, you’ll never replace the first cipher you choose!

#### My substitution cipher decryption has an ok but incorrect result? The score stops improving at some point.

There’s again a few possible causes, but remember that we want to start the hill-climbing algorithm from scratch on each of the 20 iterations. When you’re checking that a swap “improves” the decryption, you should compare the scores of the original substitution key and the swapped substitution key. You shouldn’t compare against the best overall score from previous hill-climbs, since it’s unlikely that we’ve found a better key yet.

Additionally, make sure to “reset” or throw away swaps that don’t improve the decryption. We always want to try to improve on the best-so-far in the current hill climbing iteration. If we continue swapping from a cipher that got a lower score, we’re moving further away from a correct decryption.

#### My substitution cipher decryption is really slow or running forever?

One possible cause might be calling applySubstitutionCipher, getScore, or clean too many times. In your decryption algorithm, you might be recomputing the same value over and over. Try to rewrite your code so that this only gets called when necessary!

### Task: Substitution Cipher on Files

#### I get “use of undeclared identifier” when I try to write a new helper function?

This is an annoying C++ thing, where we have to “declare” that functions exist before we use them. In most real-world situations, we declare functions inside the “header files” and #include the headers, but for this helper function, you’ll want to put a function declaration near the top of ciphers.cpp.

A function declaration is a line of code like void rot(vector<string>& strings, int amount); – it says that this function exists, and will be implemented elsewhere. This is the same as the first line of the function definition.

#### My console decryption works, but my file decryption produces gibberish?

One thing that could cause this is trying to decode each line of the file by itself. The whole file is encrypted with a single key, so the key should be the same for all lines. If you get different keys for each line, then the decoded text will definitely be wrong in some places. Additionally, using the entire text ensures that our scoring takes into account the likelihoods of all words in the cryptogram (versus those in just one line), which makes it more likely that you find the right key.

#### My file decryption looks like it produces the right file, but the test fails on Gradescope?

One possibility is that you’re not adding a newline at the end of every line, including the last one.

Another possibility is that you’re introducing an extra newline at the end of the file by reading an extra line from the input. The main way we’ve seen this happen is due to using while(!infile.eof()). The eof flag in a stream isn’t actually set until an attempted read fails. If the last line in an input file ends with a newline, getline on that line won’t set eof! Instead, the next call – which reads nothing – will set eof. Instead, you should use the pattern introduced in lecture: while(getline(infile, line)).