For this project, we will be writing a class to handle super-secret intelligence. There will be two parts to this project. The first part will be writing some functions that can redact messages so that we can safely share them with the public. The second part will be writing a self-destructing message that can only be viewed a specifed number of times before hiding the information forever.
There are 6 files in the folder "secret_messages":

- redact.h - This header file should be modified to act as the header for redact.cpp and contain the function declarations for the three redact functions.
- redact.cpp - This is the source file for the the redact function definitions.
- test_redact.cpp - This is a file that tests your redact functions. Feel free to modify it for testing purposes.
- self_destructing.cpp - This is source file for the functions and methods of the SelfDestructingMessage class. All function and method definitions for the class must be in this file.
- self_destructing.h - This is the header file for self_destructing.cpp.
- test_self_destructing.cpp - This is a file that tests SelfDestructingMessage class. You may alter this file for your testing purposes.

The test_redact.cpp and test_self_destructing.cpp shows how the functions and classes are used, but here is a quick rundown of the functionality you need to provide:

- A function named "redact_all_chars" that takes a const ref string and returns a string that is the similar to the argument string, but only has octothorpes ('#') instead of the original characters.
- A function named "redact_alphabet_digits" that takes a const ref string and returns a string that is the similar to the argument string, but only has octothorpes ('#') instead of the original characters if those characters were letters or digits (a-zA-Z0-9).
- A function named "redact_words" that takes two arguments: a const ref string (input) and a const ref vector of strings (words_to_redact). It returns a string that is the similar to the argument string, but only has octothorpes ('#') instead of the original characters if those characters composed a word in words_to_redact. Note, the matching string could be inside another word (univer###y).
- You need to write a class called "SelfDestructingMessage":
    - It has two constructors. The first takes a vector of strings (messages) and a long (number_of_allowed_views). The messages are a list of secret messages, and the number_of_allowed_views is how many times each individual message can be viewed. There is also a default construction (no messages, number_of_allowed_views is zero).
    - There is a "size" method, which returns the number of messages (regardless of number of views remaining).
    - There is a "get_redacted" method that returns a vector of strings made up of each message in redacted form. Redact all alphabetic and digit characters.
    - There is a "number_of_views_remaining" method that takes a single argument, the message index. It returns the number of views remaining for that specific message. Remember, each message starts with the number_of_allowed_views.
    - You can see the actual message using the [] operator which takes a size_t argument (index). It should return a const reference to the message string that was indexed. When a message is accessed by [], be sure to decrement the

number_of_views_remaining for that message.

    - If the index is not a legal index for a message (too large), throw a out_of_range exception.
    - If the access is to a message with no remaining views, thow a invalid_argument exception.
  - The class should allow the << operator. Each line of the message (in redacted form) is prefixed by the number of views remaining, and should be inserted to the ostream.
  - If a class is copied via a copy construction, it should transfer all the views remaining to the copy, and zero out the original. (This is to protect from trying to get more views by making copies of the object).
  - If the class is assigned, it should transfer the views remaining to the left-hand-side like with the copy constructor. This is also to prevent copy abuse.
  - The class should also implement the >> operator. It should extract a line from the istream and append it to the message list. Number of views remaining starts at number_of_allowed_views.
  - There is a method named "add_array_of_lines" that takes two arguments: an array of strings (messages to be added) and a long (the size of the array). Each string should be added to the list of messages (number of views remaining starts at number_of_allowed_views).