# General Statement

In this assignment, we simulate sending and receiving telegrams. A customer comes into the office and the program asks whether the user wants to send or pick up a telegram. If it’s “send”, the user is prompted to enter the message and the name of the recipient. The program writes the message to a file (the telegram) whose name is the name of the recipient. If the user wants to “pick up”, the user is prompted for the name of the sender and the program attempts to open a file with that name. If the file is there, the program reads the file and displays its contents. If the file is not there, the program informs the user (displays) that there is no message from (and gives the name of the sender).

## Design

This is the Pencil and Paper step; it is the Top Level Design. Note: the arrangement of this outline is **not** the same as the structure of the program! *Pay attention to the specifications*!

1. Set up the simulation
2. While there are customers
   1. Ask whether customer is there to send or pick up
   2. Get the name of the recipient/sender
      1. Send
         1. get the message
         2. write the message to a file
      2. Pick Up
         1. open the message file
            1. display the message

## Elaboration:

The **Main Program** consists of a single While loop. The condition that determines whether or not to stay in the loop (“there are customers”) is determined by a question to the user (“Are there more customers? (y/n)”). Inside the loop, the user is asked whether the purpose is to send or pick up a telegram (a numbered menu is the easiest way to ask this).

Based on the answer, the main program calls either **Function send( )** or **Function** **pickUp( )**. But first, the program asks the user for a name (either to whom or from whom). The program then calls the appropriate function, passing the name as a string parameter. Since each function returns a string value, the main program assigns the function “call” to a string that it will display.

Each function takes the input parameter, appends “.txt” to it, and opens it for reading (**Function** **pickUp( )** ) or writing (**Function** **send( )** ) using an appropriately named file object (e.g., “inFile” or “outFile”).

**Function pickUp( )** reads the entire file into a string and returns the string as the message received. **Function send( )** asks the user to type a message. It then writes that input string to its open file and returns a string confirming “message sent.”

**Each function closes its file object before returning.**

## Technicalities

1. Start with the *Template for A Pseudocode Program with a Single Function*.
   1. **Do not delete any of the comments or lines in the template**
2. Observe indentation and other style conventions
3. Use ***Turning Algorithm Design into Pseudocode*** as your guide
   1. Paste the Pseudocode into IDLE and turn all the code into comments
   2. Beneath each pseudocode command, insert the appropriate Python command that does the same thing
   3. Use the document **PSEUDOCODE – Python EQUIVALENTS** to translate Pseudocode to Python
4. Execute, debug, and revise the program until it works according to specifications
   1. **Revise the pseudocode** to match any changes you need to make to the Python code
5. Use ***Using Pseudocode to write Real Python Code*** as your guide
   1. Remove unnecessary comments made out of pseudocode statements
   2. Retain relevant comments that were in the pseudocode
   3. Add/retain/refine relevant comments from the testing and revision process
   4. Use style conventions to maximize code clarity and readability
6. **Save and submit** this cleaned-up version of the program.
7. **Save and submit** the revised pseudocode, which is now capable of producing a working program if you converted it to another language