Assignment 2
TCP Secure Text and Picture File Transfer Application [Part I]
(Due: Midnight on Saturday, December 16th, 2023)

Assignment Setup:

Create a project in Eclipse called `A2`. Download the file `A2.zip` from E-Learning. Extract the zipped files into the `A2` project. You will notice that there are the following files:

1- `A2_template.py`: This is your solution file. You need to rename the file to `A2.py`. Also, you need to enter your credentials on top of the file.
2- `A2_test.py`: This is your testing file. You can use the given functions to test your solution. This file uses multithreading which starts the server and clients on different threads.
3- There are three output files: `A2_output_basic.txt`, `A2_output_client.txt` and `A2_output.server.txt`. Your output should exactly match the given output. Do not edit these files.
4- There are three files to store your output: `A2_student_basic.txt`, `A2_student_client.txt` and `A2_student.server.txt`. The testing file will tunnel all of your output to these three files. You do not need to open or close the above three files.
5- There are six input files. These files will be used for testing. The client will use these files for upload: `sample1.txt`, `sample2.html`, `sample3.rtf`, `sample4.png`, `sample5.jpg`, `sample7.pdf`.
6- When the server downloads these files, they will be stored in the following files: `sample1_copy.txt`, `sample2_copy.html`, `sample3_copy.rtf`, `sample4_copy.png`, `sample5_copy.jpg`, `sample4_copy_copy.png`

Make sure that all your files are on the same directory level within the project.

Submission:

When you are done, you only need to submit one file: `A2.py` . Do not submit any other file. Failing to submit on time will result in 50% penalty. You have only 24 hours period for late submission.

Assignment Policies:

1- You can only import the socket library. You cannot import any other library.
2- You are not allowed to use `print` statements. (Yes! You read that right!). Instead, every function will receive an argument which is a filename. You need to use the command `write(filename,<msg>)` for your outputs. When your program is executed, nothing should be seen on `stdout` except the following:

```
Starting Testing
Task 1 Testing Complete
Task 2 Testing Complete
Task 3 Testing Complete
Task 4 Testing Complete
Task 5 Testing Complete
Task 6 Testing Complete
Task 7 Testing Complete
Task 8 Testing Complete
Task 9 Testing Complete
Task 10 Testing Complete
Task 11 Testing Complete
Task 12 Testing Complete
```

3- You can write your own private functions. However, you can't have more than three private functions for this assignment. Every private function should start with an underscore, and should be placed at the bottom of the `A2.py` file.

Assignment Technical Specifications:

This assignment aims at building a client-server application that uses TCP IPv4 socket programming. The assignment is designed to use structured programming approach (other solutions may use the object-oriented programming approach). The server uses queues for serving the client (other solutions may use multithreading).

Overview:

The objective of this assignment is to design a secure file transfer application. The application will focus on transferring text and picture files of small and medium sizes. In part I, the focus will be on building a functioning application, while the "security" consideration will be the emphasis of Part II.

The protocol is called: "Secure Text and Picture File Transfer Protocol (STPFTP). However, for short, we will call it ***STP***.

The assignment only requires basic skills of socket programming. However, it does require good understanding of the Python language.

General Specifications:

1- Both ends will communication using 'utf-8' encoding, which is defined by the ENCODING constant.
2- Both client and server will run on the localhost. The server will be using port 6330. This is defined by the SERVER_ADDRESS constant.
3- The application supports only six file extensions: txt, rtf, html, png, jpg and gif. These are defined under TEXT_EXTENSIONS and PIC_EXTENSIONS.
4- The receiving buffer at both ends should be equal to BUFFER which is 64. However, when downloading or uploading the file, the size should be 128 which is defined by the BLOCK_SIZE.

The STP Protocol:

1- The client connects to the server.
2- The client sends a list of configuration commands. Each command is formatted as: `$command:value$`.
3- When the client completes sending all commands, it sends `<configuration_complete>`.
4- The server receives and analyzes the configuration commands.
    a. If they are valid, it sends `<configuration_approved>`.
    b. If they are invalid, it sends an error message and terminates.
5- The client receives the server feedback on the configuration.
    a. If it is approved, it starts uploading the file.
    b. If it is rejected, it terminates the connection.
6- The client uploads file by sending the data in blocks. The file is processed in text format or binary format for picture files.
7- The server downloads the sent file.
8- When the server completes downloading, the message: `<download_complete>` is sent and it terminates the client connection.
9- When the client receives the feedback, it terminates the connection.

Implementation:

The template file provides the prototypes for all required functions. Some of these functions are used by the server or client, while others are common between the two.

The template file has extensive docstrings to guide your implementation. This includes defining the input arguments, return values, main tasks, errors, exceptions and dependencies (functions that it calls). Use the docstrings along with the output files to guide your implementation.