Network Programming
Assignment 2

Submission Instruction:
==========
- Due date: April 30, 2022
- File format: Compress the following files in a zip format
    (1) solution.py (2) Execution results (file name and format: output.pdf).
- zip file name: {student ID}.zip (e.g., 123456789.py)
- Example File tree
    - 123456789.zip
        - solution.py
        - output.pdf

- There will be a deduction of points (up to 50%) if you do not follow the above instruction.

Evaluation Criteria:
==========

- Your codes will be cross-checked with others to detect copying, and you will get zero points under such circumstances.
- Your codes will be evaluated based on programming style (organization, comments, readability), program function (completeness), documentation (demonstration, exception handling), etc.

Inquiries:
==========
- Send inquiries to the TA: Ankit Kumar Singh (ankit@soongsil.ac.kr).

Some Suggestions to solve the problem:

- Read the submission instructions carefully (Incorrect submission format may lead to deduction of points).
- Read textbook chapters 4, 5, and 6.
- Read problem statements 2-3 times to understand.
- Don't modify or edit the "main.py" file.
- Write your code within the specified function/method of the "solution.py" file.
- After following the given above suggestions don't solve your problem or doubt send an inquiry to TA.

Questions:
==========
[Total: 10 points]

Problem Statements:

1. **[4 points] Special bits**

You are given three integers ***L*** and ***R*** that denote a range and ***k.*** Your task is to find a number in between the range ***L*** to ***R*** (inclusive) which has a set bit count as ***k.***

If there are multiple such numbers present in the range, return the minimum among such numbers. If no number satisfies the above condition, return ***-1.***

Note

- 14 can be written as 1110 in binary and its set bit count is 3.
- 8 can be written as 1000 in binary and its set bit count is 1.

Input

The “*special_bits*” function/method of “*solution.py*” takes three integers ***L***, ***R***, and ***k*** as parameters.

Output

The “special_bits” function/method of “solution.py”, returns one integer(value of “*num*” variable) denoting the answer to the problem.

Constraints

$1 \leq L \leq R \leq 10^{18}$
$1 \leq k \leq 64$
$1 \leq T \leq 10^{5}$

Sample Input	Sample Output
1 10 1	1
1 10 2	3
1 10 3	7
1 10 4	-1

Explanation

The minimum number in range 1 - 10 with 1 set bit is 1 (0001).

The minimum number in range 1 - 10 with 2 set bit is 3 (0011).

The minimum number in range 1 - 10 with 3 set bit is 7 (0111).

No number is present in the range 1 -10 with 4 set bit count.

2. **[2 points] Toggle String**

You have been given a String ***S*** consisting of uppercase and lowercase English alphabets. You need to change the case of each alphabet in this String. That is, all the uppercase letters should be converted to lowercase and all the lowercase letters should be converted to uppercase.

Note

Don’t use any predefined function

Input
The “*toggle_string*” function of “*solution.py*” takes String ***S*** as parameters.

Output
The “toggle_string” function of “solution.py”, returns the resultant String.

Sample Input	**Sample Output**
abcDe	**ABCdE**
ANkit	**anKIT**

3. **[2 points] Network Data**

Ankit wants to design a data transfer protocol. Help ankit in completing the helper functions for his data transfer protocol.

Steps to send the data:

1. Convert the python object to a JSON string
2. Encode the JSON string to byte
3. Compress the byte data

Steps to receive the data:

1. Decompress the received byte data
2. Decode the bytes to JSON string
3. Convert JSON string to python object

Note

1. Find the corresponding helper functions in the “solution.py” file.
2. Don’t modify “send_message” and “recv_message”
3. Don’t change the name and parameters of the helper function.

4. [2 points] Securing a Socket

Complete the client function that returns the server(peer) certificate, cipher, and message received from the server.

Steps:

1. Generate a certificate and Private key file(named “ssu.crt” and “ssu.key”) using OpenSSL.
2. Combine the certificate and key file in a single PEM file(named “ssu.pem”)
3. Use the certificate and key file to secure the socket
4. Write the code in the “client” function of the “solution.py” file to connect and read the certificate, cipher, and the data received from the server.
5. Return the certificate, cipher, and data from the client function.

Note

First execute the “server.py” file to start the server. Don't modify the server.py file