2.1 Task 1: Provide a brief description of Yao's Garbled Circuits and how they function in your own words. Provide your references

Ans:
Garbled circuits can be defined has a public key encryption instead of different keys we use gates with one being the garble generator and other is the evaluator. Depending on the AND operation result, the evaluator get access to documents only when both of them agreed to share.

Ref: https://web.mit.edu/sonka89/www/papers/2017ygc.pdf

Task 2: Cloned the repo successfully and was able to run the code.

```
student@gb0401:~$ python --version
Python 3.10.12
student@gb0401:~$ git clone https://github.com/kalyancheerla/
SSS_Lab4_Garbled_Circuit_Protocol.git
Cloning into 'kalyancheerla'...
remote: Not Found
fatal: repository 'https://github.com/kalyancheerla/' not found
SSS_Lab4_Garbled_Circuit_Protocol.git: command not found
student@gb0401:~$ git clone https://github.com/kalyancheerla/ SSS_Lab4_Garbled_Circuit_Protocol.
it
Cloning into 'SSS_Lab4_Garbled_Circuit_Protocol.git'...
remote: Not Found
fatal: repository 'https://github.com/kalyancheerla/' not found
student@gb0401:~$ git clone https://github.com/kalyancheerla/SSS_Lab4_Garbled_Circuit_Protocol.g
t
Cloning into 'SSS_Lab4_Garbled_Circuit_Protocol'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 11 (delta 2), reused 7 (delta 2), pack-reused 0
Receiving objects: 100% (11/11), 5.35 KiB | 1.78 MiB/s, done.
Resolving deltas: 100% (2/2), done.
student@gb0401:~$ ls
Desktop     Downloads  Pictures  snap                              Templates
Documents   Music      Public    SSS_Lab4_Garbled_Circuit_Protocol  Videos
student@gb0401:~$ cd SSS_Lab4_Garbled_Circuit_Protocol/
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ pip install -r requirements.txt
Defaulting to user installation because normal site-packages is not writeable
Collecting pycryptodome
  Downloading pycryptodome-3.20.0-cp35-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1
B)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 14.0 MB/s eta 0:00:00
Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.20.0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python onegate_gc.py  XOR 0 0
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python onegate_gc.py  XOR 0 1
Output is 1
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python onegate_gc.py  XOR 1 0
Output is 1
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python onegate_gc.py  XOR 1 1
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$
```

Task 3:
Used the available commented code. We are able to get the desired output.

```
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python onegate_gc.py  XOR 1 1
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ gedit onegate_gc.py
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python onegate_gc.py  XOR 1 1
Encoded A (16bytes): b'\xa8L\xf9\xcb\x96\xa96|\x15\xd8\x033U\x10\x8a\xef'
Encoded B (16bytes): b'b\x18x\x8a\xc0\xa2|\x99\xea\xa0E<:vgm'
Encoded O (16bytes): b'\xeb\x0c*QXW\xbajAx\xef\xcb2\xee\xe1%'
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$
```

Task 4: Used the available commented code. We are able to get the desired output.

```
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XOR 1 1
Output Labels: [b'\x059\xfb\x1e\x8cF\xf0!\xffs\x92\x1d\\\xce\xc3h', b'3\xf0\x03\xf3~\xf73r\xd2\xd
c\xc7\xcb\x1a\xb6\r\x1d']
Garbled Table:
               16bytes - b'\r82\xab\x82\xa5bJ\xe1N}\x8d%\xb8\x1c\x83'
               16bytes - b'\xeevR`i\x85\xb5W\xdd\n\xb0@y\xc1bw'
               16bytes - b'\x1bh\xd8JTe\x01\x12 \xfe8"\xb6\xe2\x0c\xd6'
               16bytes - b'H\xfa\xe64\xff\x1ea\xc8\x93\xd0\xd9\x82\x95F\xa2\xbc'
Output is 0
```

The Table gives all the possible input combinations to match output labels in this section. An input combination and the corresponding output label are displayed on each line.

Task 5: modified the code to it's original state and on command line I changed one of the argument from XOR to AND and was able to get the desired output.

```
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ gedit andgate.py
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  AND 1 1
Output is 1
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  AND 1 0
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  AND 0 1
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  AND 0 0
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$
```

Task 6:
Added following lines of code

```
def XNOR_gate(a,b): return ~(a^b) & 1
```

and the below line in main function for arguments check

```
'XNOR' : XNOR_gate,
```

And was able to get desired output as seen in the below screenshot.

```
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ gedit andgate.py
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  0 0
Output is 1
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  0 1
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  1 0
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  1 1
Output is 1
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$
```

Task 7:
The Garbler Logic explains the encoding the received inputs, creating a Garbled Table with corresponding labels, and safely sending this data to an assessor. The evaluator uses the garbled data to do calculations, and the Garbler decodes the outcome to protect privacy all along the way.

Task 8:
After getting garbled data the code searches through entries in the Garbled Table and assessing the gate result, producing an output which is encoded.
Next returns it to the garbler as part of the Evaluator Logic of a Garbled Circuit protocol. The protocol's privacy-preserving feature guarantees that the evaluator can calculate the outcome without learning the inputs' true values.

Task 9:

The code implements a concept related to secure computation, specifically a form of what could be described as garbled circuits, using double encryption for securing the computation of logic gates (AND, OR, XOR, XNOR). The encryption cipher used for the double encryption process is AES (Advanced Encryption Standard) in ECB (Electronic Codebook) mode, as evidenced by the instantiation of the AES cipher objects within the double_encrypt and double_decrypt functions.

Yes, we can use a different symmetric cipher in place of AES. The code structure is such that the encryption and decryption mechanisms are encapsulated within the double_encrypt and double_decrypt functions, respectively.
This design allows for the substitution of the AES cipher with any other symmetric cipher like DES, 3DES, ChaCha20, or Blowfish as long as the new cipher supports the same interface for initialization, encryption, and decryption. The key consideration when substituting AES with another cipher is to ensure the security properties of the alternative cipher meet the requirements of the application. Also, compatibility regarding key sizes, block sizes should be taken into account.

As for changing the order of encryption from cipher2.encrypt(cipher1.encrypt(data)) to cipher1.encrypt(cipher2.encrypt(data)), it's important to understand that encryption and decryption operations are inverse operations. If the order of encryption operations is reversed, the order of decryption operations must also be reversed to correctly decrypt the data. Thus, if the data is encrypted first with cipher2 and then with cipher1, you must first decrypt with cipher1 and then with cipher2 to obtain the original data. Therefore, for the changed encryption order:

```
# Changed encryption order
return cipher1.encrypt(cipher2.encrypt(data))
```

The corresponding decryption order should be:

```
# Corresponding decryption order for the changed encryption
return cipher2.decrypt(cipher1.decrypt(encrypted_data))
```

This reversal ensures that the first operation to encrypt is the last to decrypt.

Task 10:
The code primarily illustrates a two-party setup in the context of Secure Function Evaluation (SFE), specifically within the garbled circuit framework. The two parties are implicitly represented as follows:

The Garbler: This party is responsible for generating the garbled circuit, which includes creating garbled tables for logic gates and encrypting the outputs based on the inputs. The garbler's operations are encapsulated in functions like generate_labels, garble_gate, and double_encrypt.

The Evaluator: This party evaluates the garbled circuit with their own inputs without learning anything about the garbler's inputs. This is done through the evaluate_garbled_gate and double_decrypt functions.

Yes, we can update the code. To extend this setup to support Multi-Party Computation (MPC) for more than two parties, a few conceptual and structural adjustments are necessary. MPC involves multiple parties computing a function over their inputs while keeping those inputs private from each other. There are various approaches to MPC, each with its own set of protocols and techniques, such as secret sharing, homomorphic encryption, and more sophisticated garbled circuits.

Steps to adapt to support MPC:

Step 1: Identify the Computation Model
First, we need to determine the MPC model to use. Since the current code is based on garbled circuits, we can look into extending it with techniques that allow for multi-party garbled circuits or explore other models like homomorphic encryption that designed for multiple parties.
Step 2: Redesign the Input and Encryption Mechanism
Step 3: Adapt the Garbling Scheme for Multi-Party Evaluation
Step 5: Result

Task 11: I modified the exicting code for according to the given circuit gate. So, for the program to run we will be using only values not the gate type. So we will get the same result which gate type we give. We can modify even the gate type to have two gate types given in command line if we want to.

```
TypeError: list indices must be integers or slices, not str
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ gedit andgate.py
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  OR  0 0 0
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  0 0 0
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  0 0 1
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  0 1 1
Output is 1
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  1 1 1
Output is 1
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  1 1 0
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  1 0 0
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  1 0 1
Output is 1
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  1 1 0
Output is 0
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$ python andgate.py  XNOR  0 1 1
Output is 1
student@gb0401:~/SSS_Lab4_Garbled_Circuit_Protocol$
```