Homework1: Tree Christmas tree & Diffie-Hellman key exchange cracker

Deadline & Submission

MUST READ

- Submit your `hw1.py` file on **Ceiba**.
- Deadline : Oct. 23 23:59
- No late submission.
- Do not copy others code. (0 scores for punishment)

Introduction

You need to be familiar with the following topics to get the homework done:

- Basic python syntax
- Basic input and output
- Create python function
- Usage of for and while loop

In this homework, you will need to design a Christmas tree generator and a function to crack Diffie-Hellman key exchange protocol.

Christmas Tree Generator

In this part, you will need to design a Christmas tree generator that you can enter a number and generate Christmas trees based on the entered numbers.

If you don't know what the output Christmas tree looks like, you can use **hw1_example.py** to generate a sample Christmas tree and your generator in your homework must output the same result as sample.

The usage of **hw1_example.py** is as follows:

If you want to produce a 3-layer Christmas tree, you need to enter the following command in the command line environment(ubuntu):

```
$ python3 hw1_example.py 3
               /*\
              /***\
             /*****\
          /***********\
         /*************\
        /***************\
       /*****************\
    /***********************\
   /*************************\
  /************|||************\
 /*************|||*************\
/**************|||**************\
```

If you want to produce a 5-layer Christmas tree, you need to enter the following command in the command line environment(ubuntu):

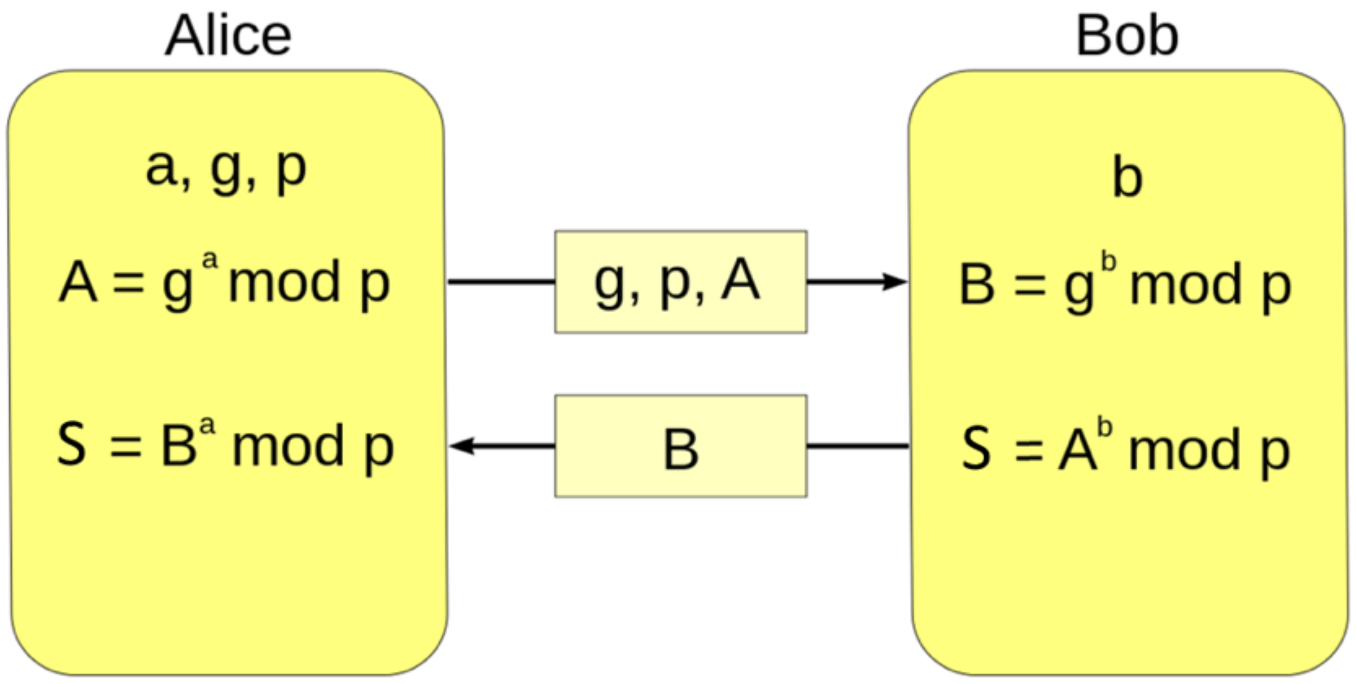
```
$ python3 hw1_example.py 5
                                  /*\
                                 /***\
                                /*****\
                             /***********\
                            /*************\
                           /***************\
                          /*****************\
                       /***********************\
                      /*************************\
                     /***************************\
                    /*****************************\
                   /*******************************\
               /***************************************\
              /*****************************************\
             /*******************************************\
            /*********************************************\
           /***********************************************\
          /*************************************************\
      /*********************************************************\
     /***********************************************************\
    /****************************|||||****************************\
   /*****************************|||||*****************************\
  /******************************|||@|******************************\
 /*******************************|||||*******************************\
/********************************|||||********************************\
```

The number entered may be very large, and your function must be able to generate a Christmas tree with the correct specifications.

Diffie-Hellman key exchange cracker

In this part, you need to implement a function to crack Diffie-Hellman key exchange protocol.

In cryptography, there is a security protocol called Diffie-Hellman key exchange, which allows both parties to establish a key through an insecure channel without any prior information. Then, they are able to encrypt their message with this key.



$$S = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$$

Suppose there are two people, Alice and Bob, who need to send messages between each other.
First of all, they must have to agree on a prime number p and base number g, and make it public. Next, Alice and Bob will choose a private number respectively (sometimes called a private key) a and b, where $a, b \in \{1, 2, \ldots, p-1\}$, and keep them so that others will not know.
So Alice doesn't know what b is, and Bob doesn't know what a is.
They will use a and b to calculate their respective public keys A and B as the following method:

$$A = g^a \; mod \; p \quad and \quad B = g^b \; mod \; p$$

And make these two key public for everyone to know. Here **mod** is the operation of taking the remainder(餘數).
After obtaining other side's public key, both perform the following calculation respectively:

- Alice：

$$B^a \; mod \; p = (g^b)^a \; mod \; p = K$$

- Bob：

$$A^b \; mod \; p = (g^a)^b \; mod \; p = K$$

In this way, they can calculate a same number, which is the share key between them, and finally they can use this key to encrypt all messages to be sent(example: can be Caesar cipher).

For example, if they choose $p = 11$、$g = 3$, they can find K as following steps:

- Choose their private key a and b eg:
  Alice choose $a = 4$, Bob choose $b = 2$.
- Calculate their public key A and B eg:
  - Alice:
    $$A = 3^4 \, mod \, 11 = 4$$
  - Bob:
    $$B = 3^2 \, mod \, 11 = 9$$
- Calculating common key K:
  - Alice:
    $$K = 9^4 \, mod \, 11 = 5$$
  - Bob:
    $$K = 4^2 \, mod \, 11 = 5$$
- Finally, they get their common $K = 5$

As an attacker, you want to crack the private keys a and b belong to Alice and Bob, so that the common secret K can be calculated and crack the messages sent between them.
In other words, if I want to find out how many times g will become A and how many times g will become B.
This problem is called ***Discrete logarithm*** in cryptography.
In practice, p will be a very large prime number, so that it takes several years to calculate a and b or is almost impossible to calculate.
But in this assignment, we limit $2 \leq p < 100$ so that the calculation can be completed in a short time.

Your function have to input 4 positive integers p, g, A and B as parameters, where $2 \leq p < 100$, $2 \leq g \leq 50$, $2 \leq A \leq p - 1$, $2 \leq B \leq p - 1$, and return private key a , b and common key K.

Requirements

Given the template code provided by TA, you need to fulfill all the methods and functions in the code. The score/point for each method or function is explained in the docstring.

MUST READ

- You cannot use any third-party package such as numpy, pandas, and etc.
- You can only use python primitive types and statements to solve the problem.
- Do not copy others code. (0 scores for punishment)

Expected Execution Result

In this assignment, you must finish following functions as requirement.

TA will test your function with randomly input two positive integer as patameters.

print_level_two_tree()

You have to print out a level two tree in this part.
If you don't have what a level two tree look like, you can use **hw1_example.py** to generate a level two tree.
It will look like this:

```
print_level_two_tree()
```

```
          /*\
         /***\
        /*****\
     /***********\
    /*************\
   /***************\
  /********|********\
```

find_middle(n)

TA will test your function with randomly input a positive integer as patameters.
In this part, you have to find the middle line of the tree and return the distance from the middle line to left end.

For example, if the input number is 2, you have to find the middle line of a level two tree and return the distance to left end that is 10.

```
print(find_middle(2))
```

```
10
```

```
print(find_middle(3))
```

```
17
```

print_level_n_tree(n)

In this part, you have to print the level n tree which n is a integer.
You can use 'hw1_example.py' to take a look what a level n tree looks like.
For example, if the input number is 2, you have to print out a level two tree, and it will looks like:

```
print_level_n_tree(2)
```

```
        /*\
       /***\
      /*****\
   /***********\
  /*************\
 /***************\
/********|********\
```

```
print_level_n_tree(5)
```

```
                                  /*\
                                 /***\
                                /*****\
                             /***********\
                            /*************\
                           /***************\
                          /*****************\
                       /***********************\
                      /*************************\
                     /***************************\
                    /*****************************\
                   /*******************************\
               /***************************************\
              /*****************************************\
             /*******************************************\
            /*********************************************\
           /***********************************************\
          /*************************************************\
      /*********************************************************\
     /***********************************************************\
    /****************************|||||****************************\
   /*****************************|||||*****************************\
  /******************************|||@|******************************\
 /*******************************|||||*******************************\
/********************************|||||********************************\
```

Diffie_Hellman_cracker(p, g, A, B)

In this part, you need to implement a function which can enter four positive integers as parameters and find out the private key a ,b and common number K then return those numbers.

```
print(Diffie_Hellman_cracker(11,3,4,9))
```

```
(4,2,5)
```