Homework 2

Moodle Submission Deadline: 2023/03/29 (Wed) 23:59

[Bring the printed papers to the class on **03/30** (Thu)]

Problem 1. Python Basics, Conditionals, Loops (hw2_p1.docx)

Problem 1-1. Given the initial statements:

```
s1 = "spam"
s2 = "ni!"
```

Show the result of evaluating each of the following string expressions.

(a) `"The Knights who say," + s2`

(b) `3 * s1 + 2 * s2`

(c) `s1[1]`

(d) `s1[1:3]`

(e) `s1[2]+ s2[:2]`

(f) `s1 + s2[-1]`

(g) `s2[len(s2)//2]`

Problem 1-2. Given the same initial statements as in problem 1-1, show a Python expression that could construct each of the following results by performing string operations on `s1` and `s2`.

(a) `"NI"`

(b) `"ni!spamni!"`

(c) `"SpamNi! SpamNi! SpamNi!"`

(d) `"span"`

(e) `"spm"`

Problem 1-3. Show the string that would result from each of the following string formatting operations. If the operation is not legal, explain why and correct the formatting operation.

(a) `"Looks like %s and %s for breakfast" % ("spam", "eggs")`

(b) `"There is %d %s %d %s" % (1, "spam", 4, "you")`

(c) `"Hello %s" % ("Suzie", "Programmer")`

(d) `"%0.2f %0.2f" % (2.3, 2.3468)`

(e) `"%7.5f %7.5f" % (2.3, 2.3468)`

(f) `"Time left %02d:%05.2f" % (1, 37.374)`

(g) `"%3d" % ("14")`

Problem 1-4. Briefly describe the output of each small Python program below. Please have full understanding for (i) and (j). Note that for (i), please try to use different `a` and `b`, and observe the resulting `a`. For (j), please try to use different `n`, and observe what will be printed out.

(a)
```
x = 5
y = 3
if x >= y:
    x = x - 2
print(x)
```

(b)
```
tc = 100
tf = (9/5) * tc + 32
print(tf)
```

(c)
```
x = 0
while x < 5:
    x = x + 1
print(x)
```

(d)
```
x = 1
i = 1
while x <= 5:
    x = x * i
    i = i+1
    print(x)
```

(e)
```
x = 0
while x < 6:
    if x % 2 == 0:
        print('even', x)
    else:
        print('odd', x)
    x = x + 1
```

(f)
```
i = 0
while i < 6:
    j = 0
    while j < i:
        print("*")
        j = j + 1
    i = i + 1
    print()
```

(g)
```
score = 40
while score > 1:
    score = score/2 - 1
print(score)
```

(h)
```
x = 2
y = 7
while x < y:
    x = 2 * x
print(x)
```

(i)
```
a, b = 63, 105
while b:
    a, b = b, a % b
print(a)
```

(j)
```
n = 21
while n != 1:
    print(n, end=", ")
    if n % 2 == 0:
        n = n // 2
    else:
        n = n * 3 + 1
print(n, end=".\n")
```

Problem 1-5. Briefly describe the output of each small Python program below.

(a)

```
x = 7
y = 8
if x < 7 or x <= 10 and y > 8:
    print("ugh")
else:
    print("yuck")
```

(b)

```
phrase = "python"
vowels = "aeiou"
count = 0
while (not phrase[count] in vowels):
    count = count + 1
print(count)
```

(c)

```
if 'alpha' < 'zebra':
    print('alpha < zebra')
elif 'alpha' > 'zebra':
    print('alpha > zebra')
elif 'alpha' == 'zebra':
    print('alpha == zebra')
else:
    print('none of the above')
```

Problem 1-6. Who is the True Thief?

Four potential criminals were suspected to be the thief, and detained by polices. Their IDs are 1, 2, 3, and 4. It is sure that one of them is the true criminal. During the questioning and interrogation, each of them provides one sentence, as listed below.

```
1 said he is not the thief.
2 said 3 is the thief.
3 said 4 is the thief.
4 said 3 is a liar.
```

Evidences tell the police that three of these four sentences are true, and one is false. Please complete the following code in order to judge who is the true thief. Your task is to use Boolean operators (and, or, not) and relational operators (==, !=) with some expressions to find the true thief.

Sample Input/Output:

```
The true thief is 3.
```

Hint: 試著使用 while 迴圈嘗試判斷每個人是小偷的情況，並使用 if 判斷有幾個人是說真話

```
thief = 1
while thief <= 4:
    # single or multiple lines
    if             :
        print('The thief is', thief)
    thief = thief + 1
```

Please write down the complete code in your answer sheet. The blank part can be one or multiple lines. Note that you are not necessary to follow the above code if you have a different idea about how to solve this problem.

Problem 2: Finding Perfect Numbers (hw2_p2.py)

Write a Python program to find Perfect numbers from 2 to n, where n is input by the user. According to Wikipedia : In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors (一個數恰好等於它的因數之和), that is, the sum of its positive divisors excluding the number itself. Equivalently, a perfect number is a number that is half the sum of all of its positive divisors (including itself). For example, the first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and 1 + 2 + 3 = 6. Equivalently, the number 6 is equal to half the sum of all its positive divisors: (1 + 2 + 3 + 6) / 2 = 6. The next perfect number is 28 = 1 + 2 + 4 + 7 + 14.

Note that in your program (hw2_p2.py), you are required to write the comments to describe the meaning of each part.

Sample Input and Output

```
C:\Python35\workspace\2023計算機概論>python hw2_p2.py
Input the range number: 10000
Perfect numbers:
6
28
496
8128
```

Problem 3: Calendar Generation (hw2_p3.py)

Your task in this problem is to write a program that can generate the calendar for the input year and month. Users are allowed to provide any year and any month as the input. This problem contains some challenges. The first one is to determine the day of the week (指定日期是在一周中的星期幾). Please feel free to see the Wikipedia page to understand how to find the day of the week: https://en.wikipedia.org/wiki/Determination_of_the_day_of_the_week
or just google the equation of determining the data of the week.

The second challenge is to print the dates of the input year and month in a particular format. You are asked to follow the format, as shown below. The last challenge comes in leap years (when there are 29 days in February). In leap years, the 29th of February is a valid date, and every day after that is one day later in the year. You can again refer to the Wikipedia page for the rules of leap year: https://en.wikipedia.org/wiki/Leap_year
註：此題規定不能使用任何 Python 套件，例如 calendar 套件。

Note that in your program (hw2_p2.py), you are required to write the comments to describe the meaning of each part.

Sample Input and Output

```
C:\Python35\workspace\2023計算機概論>python hw2_p3.py
Please input Year: 2017
Please input Month: 5
Sun Mon Tue Wed Thu Fri Sat
    01  02  03  04  05  06
07  08  09  10  11  12  13
14  15  16  17  18  19  20
21  22  23  24  25  26  27
28  29  30  31

C:\Python35\workspace\2023計算機概論>python hw2_p3.py
Please input Year: 2016
Please input Month: 10
Sun Mon Tue Wed Thu Fri Sat
                    01
02  03  04  05  06  07  08
09  10  11  12  13  14  15
16  17  18  19  20  21  22
23  24  25  26  27  28  29
30  31

C:\Python35\workspace\2023計算機概論>python hw2_p3.py
Please input Year: 2012
Please input Month: 2
Sun Mon Tue Wed Thu Fri Sat
            01  02  03  04
05  06  07  08  09  10  11
12  13  14  15  16  17  18
19  20  21  22  23  24  25
26  27  28  29

C:\Python35\workspace\2023計算機概論>
```

Problem 4: Christmas Celebration (hw2_p4.py)

Printing symbols in certain patterns is a popular practice with if-else and loops, here we provide an “elevated” version for you to solve.

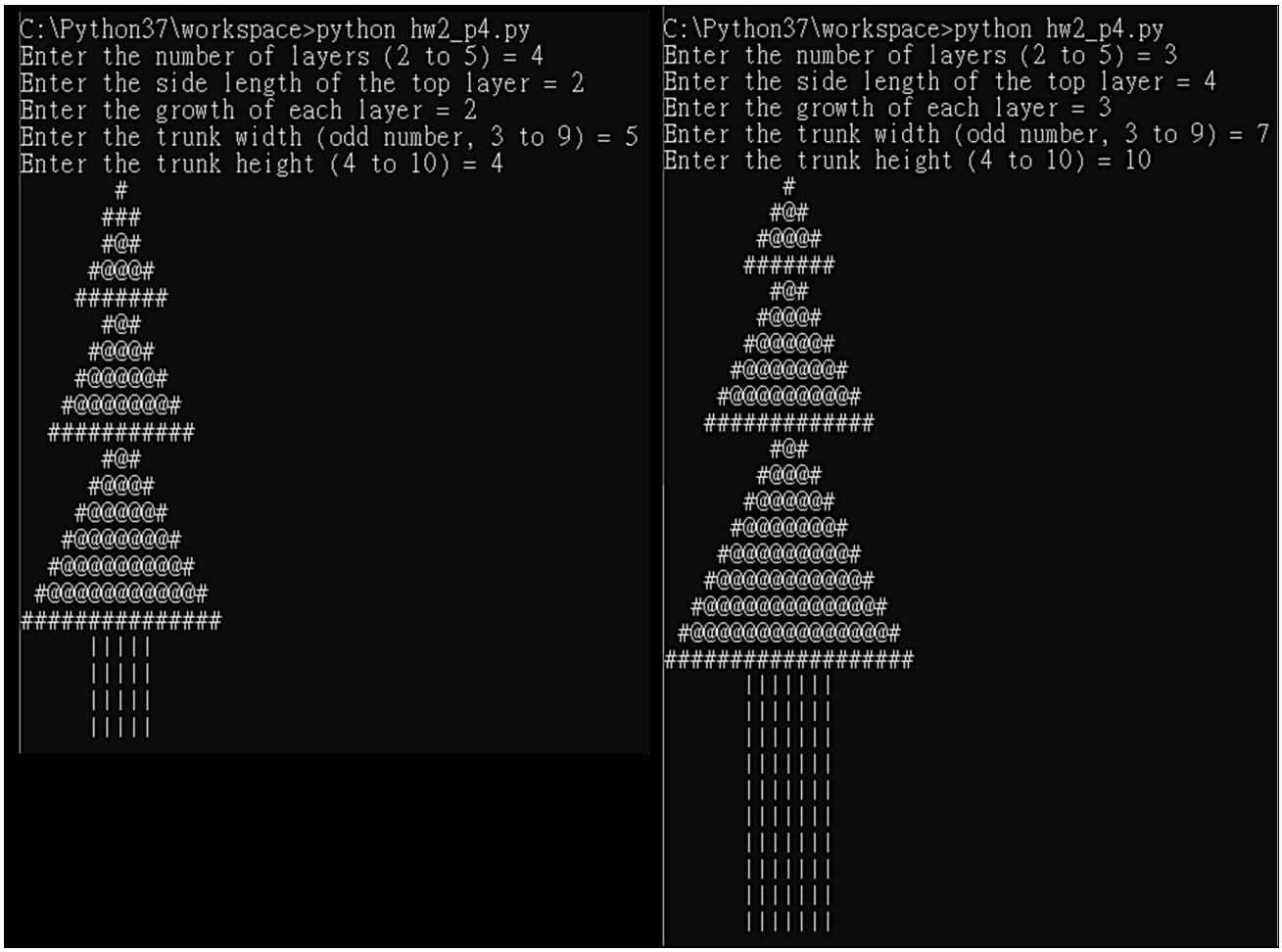
To print a Christmas tree, first you have to know some basic descriptions. A Christmas Tree consists of multiple layers of solid isosceles triangles, and the side lengths of each outer-triangle ascends as the triangle’s position descends. The bottom of each upper-triangle is overlapped with the top of the lower-triangle. Below the bottom-triangle, you should print a solid rectangle as the trunk of this Christmas Tree. Further definitions of printing the Christmas Tree are as follows:

1. The number of Christmas Tree layers is defined between 2 and 5 (included).
2. The side length of the top-triangle is defined between 3 and 6 (included).
3. The side length of each lower-triangle has to excess the side length of the upper-triangle by 1 to 5 (included).
4. The width of the trunk has to be an odd number between 3 to 9 (included).
5. The height of the trunk has to be a number between 4 to 10 (included).
6. Use ‘#’ as the sides of each triangle, and use ‘@’ as the internal part of each triangle.
7. Use ‘|’ to represent the trunk part of the Christmas Tree.

< Terms explained >

1. Upper-triangle and lower-triangle
   For each pair of triangles, the upper one is the upper-triangle, and the lower one is the lower-triangle. Remember, this does not necessarily indicate the top or the bottom one.
2. Top-triangle
   The triangle on the top of the Christmas Tree, this refers to the only triangle on the top.
3. Bottom-triangle
   This indicates the triangle on the bottom of the tree, which connects with the trunk part.
4. Side length
   The side length of a triangle is the number of points of the sides of each triangle.

Sample Input and Output



Problem 5: Encrypt!! (hw2_p5_1.py, hw2_p5_2.py, hw2_p5_3.py)

Someone wants to send a message to her parents. But she is afraid state apparatus will tap the message. Therefore, she develops a way to encrypt the message.

Frist of all, we need to transform all characters in each word to ASCII code (except space), and transform back to word after encrypting them. You can use `ord()` for retrieving an ASCII code from character, and use `chr()` for retrieving a corresponding character from ASCII code.

P.S. For more about ASCII, please visit: https://zh.m.wikipedia.org/wiki/ASCII

Second, we apply two well-known cipher methods: **Caesar Cipher** and **Affine Cipher**, to encrypt the message. Assume in the following equations, $\mathbf{y}$ represents "after encryption" and $\mathbf{x}$ represents "before encryption". $\mathbf{k}$ represents the key used for Caesar Cipher, and $(\mathbf{a}, \mathbf{b})$ represents the key pair used for Affine Cipher.

Caesar Cipher uses an encryption key $\mathbf{k}$ to shift all the words with the following formula:

$$y = x + \mathbf{k}$$

And Affine Cipher uses the key $(\mathbf{a}, \mathbf{b})$ to encrypt all of the words with the formula:

$$y = \mathbf{a}x + \mathbf{b}$$

For example: If we want to encrypt the word 'W', and $\mathbf{k}$ is given by 12, $(\mathbf{a}, \mathbf{b})$ is given by (2,4). Then we can first acquire the corresponding ASCII code: `ord('W') = 87`. And by Caesar Cipher, we perform the 1st - stage encryption with `87 + 12 = 99`. Next, we perform the 2nd – stage encryption by Affine Cipher with `99 * 2 + 4 = 202`. After going through the two stages of encryption, we have to check if the range is valid for ASCII – character transformation. We can see that 202 is out of range of upper-cased alphabets (A (65) ~ Z (90)), therefore we have to transfer back into range. Thus we perform further calculations by `((202-65)%26)+65 = 72`, which is valid since the ASCII code is between 65 (A) and 90(Z). With the ASCII code 72, we can retrieve the corresponding character with `chr(72) = 'H'`, and thus get the outcome : 'H'.

Now, you have 3 parts to clear this task. First, use Fibonacci Sequence (**hw2_p5_1.py**) to find out the key $\mathbf{k}$ used for Caesar Cipher. Second, calculate the length of the longest Palindromic substring (**hw2_p5_2.py**) for two different strings to obtain the key-pair $(\mathbf{a}, \mathbf{b})$ used for Affine Cipher. Last, you need to encrypt the message (**hw2_p5_3.py**) with the two encryption methods with the keys obtained in the previous parts.

The details of the 3 parts will be explained in the following sections respectively.

Part 1 **Fibonacci Sequence** (**hw2_p5_1.py**)

A Fibonacci sequence is a sequence of numbers where each successive number is the sum of the previous two. The classic Fibonacci begins: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ... etc. You are asked to use the `while` loop write a program that can compute the n-th Fibonacci sequence number, where n is a value input by the user. For example, if n = 0, then the result is 0. If n = 6, then the result is 8. We can write the rule of Fibonacci sequence as: $\boldsymbol{x_n = x_{n-1} + x_{n-2}}$, where x_n is the n-th Fibonacci sequence number, x_{n-1} and x_{n-2} are (n-1)-th and (n-2)-th Fibonacci numbers. The initial values are $x_0 = 0$, $x_1 = 1$. You can find the rule in the following table.

n	$x_n = x_{n-1} + x_{n-2}$	The n-th Fibonacci sequence number
0		0
1		1
2	$x_2 = x_1 + x_0 = 1 + 0 = 1$	1
3	$x_3 = x_2 + x_1 = 1 + 1 = 2$	2
4	$x_4 = x_3 + x_2 = 2 + 1 = 3$	3
5	$x_5 = x_4 + x_3 = 3 + 2 = 5$	5
6	$x_6 = x_5 + x_4 = 5 + 3 = 8$	8
n	$x_n = x_{n-1} + x_{n-2}$	x_n

The main task of this part is to retrieve the key $\mathbf{k}$ used for Caesar Cipher. For a given n, calculate the corresponding n-th number from the Fibonacci Sequence.

Sample Input and Output

```
C:\Python37\workspace>python hw2_p5_1.py
Input an integer number: 6
The 6-th Fibonacci sequence number is:8

C:\Python37\workspace>python hw2_p5_1.py
Input an integer number: 14
The 14-th Fibonacci sequence number is:377

C:\Python37\workspace>python hw2_p5_1.py
Input an integer number: 30
The 30-th Fibonacci sequence number is:832040
```

Part 2 **Longest Palindromic substring** (**hw2_p5_2.py**)

The *Longest Palindromic Substring* (LPS) is the problem of finding a **maximum-length contiguous substring** of a given string that is also a **palindrome** (回文). For example, the longest palindromic substring (LPS) of "bananas" is "anana". Here a string is said to be **palindrome** if reverse of the string is same as string. For example, "abba" is palindrome, but "abbc" is not palindrome. Note that the longest palindromic substring is not guaranteed to be unique; for example, in the string "abracadabra", there is no palindromic substring with length greater than three, but there are two palindromic substrings with length three, namely, "aca" and "ada".

Given a string `s`, your task in this problem is to write a program that can find and print out the longest palindromic substring in `s`. Here lists some examples. If the input string `s` is "babad", the LPS is either "bab" or "aba". Another example is: if the input string `s` is "cbbd", the LPS is "bb".

In this problem, you will be able to practice the conditionals (`if`, `else`), the `while` loop (and `continue`, `break`), logic operations (`and`, `or`, `not`), relational operations (>, <, >=, <=, ==, !=), string index and slicing, or built-in string functions, such as `len()`.

The main task of this part is to retrieve the key-pair **(a, b)** used for Affine Cipher. Given two strings, return the length of the longest Palindromic substring for each string. The length of the first substring will be **a** , and the length of the second substring will be **b.**

Sample Input and Output

```
C:\Python37\workspace>python hw2_p5_2.py
Enter a string: babad
Longest palindrome substring is: bab
Length is: 3

C:\Python37\workspace>python hw2_p5_2.py
Enter a string: cbbd
Longest palindrome substring is: bb
Length is: 2

C:\Python37\workspace>python hw2_p5_2.py
Enter a string: forgeekskeegfor
Longest palindrome substring is: geekskeeg
Length is: 9

C:\Python37\workspace>python hw2_p5_2.py
Enter a string: banana
Longest palindrome substring is: anana
Length is: 5
```

Part 3 Message Encryption (**hw2_p5_3.py**)

After retrieving two keys for each encryption methods in the previous parts, we now combine them with the methods to perform the final encryption.

In this problem, you will write a program which will read n to retrieve the n-th number from the Fibonacci Sequence, and store the outcome as the key for Caesar Cipher. Then, your program will also read two different strings to retrieve the longest Palindromic substring of these strings respectively. The lengths of these words will be the key-pair of Affine Cipher. Next, your program will read the plaintext(string to be encrypted), and output the encrypted text after encryption.

Some examples are listed in the table below, you can check your output with the corresponding inputs to validate your program.

Fibonacci Sequence		Longest Palindromic substring				String to encrypt	Encrypted output
n	F(n)	s_1	l_1	s_2	l_1	s_i	s_o
3	**2**	banana	**5**	detartrate	**9**	DAY	ITJ
3	**2**	allan	**4**	awkward	**5**	APPLE	AIISQ
6	**8**	tattarrattat	**12**	banana	**5**	ASSIST	KSSCSE
7	**13**	essentially	**4**	redivide	**7**	COMPUTER	CYQCWSKK
8	**21**	opposite	**4**	reversing	**5**	PYTHON	GQWACY

Note that in your program (hw2_p5_3.py), you have to print section notifications (“extract key for encrypt method” after user input and “encryption completed” before printing the final outcome).

Sample Input and Output

```
C:\Python37\workspace>python hw2_p5_3.py
The number of the requested element in Fibonacci (n) = 3
The first string for Palindromic detection (s1) = banana
The second string for Palindromic detection (s2) = detartrate
The plaintext to be encrypted: DAY
----- extract key for encypt method -----
The 3-th Fibonacci sequence number is:2
Longest palindrome substring within first string is: anana
Length is: 5
Longest palindrome substring within second string is: etartrate
Length is: 9
----- encryption completed -----
The encrypted text is: ITJ

C:\Python37\workspace>python hw2_p5_3.py
The number of the requested element in Fibonacci (n) = 6
The first string for Palindromic detection (s1) = tattarrattat
The second string for Palindromic detection (s2) = banana
The plaintext to be encrypted: ASSIST
----- extract key for encypt method -----
The 6-th Fibonacci sequence number is:8
Longest palindrome substring within first string is: tattarrattat
Length is: 12
Longest palindrome substring within second string is: anana
Length is: 5
----- encryption completed -----
The encrypted text is: KSSCSE

C:\Python37\workspace>python hw2_p5_3.py
The number of the requested element in Fibonacci (n) = 8
The first string for Palindromic detection (s1) = opposite
The second string for Palindromic detection (s2) = reversing
The plaintext to be encrypted: PYTHON
----- extract key for encypt method -----
The 8-th Fibonacci sequence number is:21
Longest palindrome substring within first string is: oppo
Length is: 4
Longest palindrome substring within second string is: rever
Length is: 5
----- encryption completed -----
The encrypted text is: GQWACY
```

Note

This is a homework for each **team**. Please submit your homework **by every team member**.

How to Submit Your Homework? [Both (1) and (2) need to be done!]

(1) Submission in NCKU Moodle

Before submitting your homework, please zip the files (**hw2_p1.docx**, **hw2_p2.py**, **hw2_p3.py**, **hw2_p4.py**, **hw2_p5_1.py**, **hw2_p5_2.py**, **hw2_p5_3.py**) in a zip file, and name the file as "學號 1_學號 2_hw2.zip". For example, if your 學號 of your team are H12345678 and H87654321, then your file name is:

"H12345678_H87654321_hw**2**.zip" or "H87654321_H12345678_hw**2**.rar"

When you zip your files, please follow the instructions provided by TA's slides to submit your file using NCKU Moodle platform http://moodle.ncku.edu.tw .

(2) Print out Your Codes and Files

Please print out your files (**hw2_p1.docx**, **hw2_p2.py**, **hw2_p3.py**, **hw2_p4.py**, **hw2_p5_1.py**, **hw2_p5_2.py**, **hw2_p5_3.py**) using A4 papers, clearly highlight which papers belong to which problems, and staple and nail these papers together. Then **bring your printed-and-nailed papers to the class on 3/30**. Please make sure your printed version is exactly the same as the submitted version in Moodle.

Have Questions about This Homework?

Please feel free to visit TAs, and ask/discuss any questions in their office hours. We will be more than happy to help you.

Final Remark

You may find that Homework 2 is more challenging than Homework 1. In fact, these are the best and basic training for your programming thinking and logical thinking (with if, elif, else and the while loops). When you are able to write Homework 2 **by yourself**, you have learned one of the most important parts of programming. Then we believe that you will be able to successfully pass the Programming Exam 2-3 and Midterm, and use programming to solve a number of real-world problems. We wish all of you can successfully submit Homework 2. ☺