Homework 2.

- The file name of your homework (**in PDF**) should be in the format: "學號-python-作業編號.pdf". For example: **00757999-python-hw2.pdf**
- Please submit your homework to Tronclass **before 23:59, November 09, 2025**

1. **(15%)** Given a nested list of numbers, matrix. Write a code snippet that processes each sublist within matrix according to the following rule: reorder the elements so that all odd numbers come first, sorted in ascending order, followed by all even numbers, sorted in descending order. Finally, print the modified matrix.

   Example:

```
# Input list
matrix = [[3, 8, 1, 5], [9, 4, 2, 7], [6, 0]]

# Your code should transform the matrix

# Expected output
# [[1, 3, 5, 8], [7, 9, 4, 2], [6, 0]]
```

```
#直到 EOF (Ctrl + Z / Ctrl + D)
matrix = []
try:
    while True:
        line = input().strip()
        if not line:
            continue
        sublist = list(map(int, line.split()))
        matrix.append(sublist)
except EOFError:
    pass
for i in range(len(matrix)):
    odds = sorted([x for x in matrix[i] if x % 2 == 1])
    evens = sorted([x for x in matrix[i] if x % 2 == 0], reverse=True)
    matrix[i] = odds + evens
print("結果：", matrix)
```

2. **(15%)** Given a list data_list and an integer k. Write a code snippet to circularly shift the elements of data_list to the right by k positions.
   Requirements:
   A. You must modify the original list directly (in-place). Do not create a new list to store the result.

   Example:

```
# Input
data_list = [1, 2, 3, 4, 5, 6, 7]
k = 3

# After your code executes

# The content of data_list is expected to become
# [5, 6, 7, 1, 2, 3, 4]
```

```python
data_list = list(map(int, input().split()))
k = int(input())
n = len(data_list)
if n > 0:
    k %= n
    data_list.reverse()
    data_list[:k] = reversed(data_list[:k])
    data_list[k:] = reversed(data_list[k:])
print(data_list)
```

3. **(20%)** Given a base template base_template = ["User", ["READ"]]. Complete the following three parts in order and explain the results.
   A. Assignment: Assign base_template to user_a. Change the name in user_a to "Alice" and add "WRITE" to its permissions list. Print both user_a and base_template, and explain why base_template was also modified.

```python
import copy
base_template = ["User", ["READ"]]

# (A)
user_a = base_template
user_a[0] = "Alice"
user_a[1].append("WRITE")

print("A - user_a:", user_a)
print("A - base_template:", base_template)

# 說明：
# 直接賦值（Assignment）只是讓 user_a 和 base_template 指向同一個物件。
# 所以改 user_a 的內容，base_template 也一起被改動。
```

   B. Shallow Copy: Create user_b by making a shallow copy of base_template (using copy.copy() or [:]). Change the name in user_b to "Bob" and add "DELETE" to its permissions list. Print both user_b and base_template, and explain why the permissions list was modified in both, but the name was not.

```python
# (B)
base_template = ["User", ["READ"]]
user_b = base_template[:]
user_b[0] = "Bob"
user_b[1].append("DELETE")

print("B - user_b:", user_b)
print("B - base_template:", base_template)

# 說明：
# 淺層複製只複製外層，內層列表還是共用。
# 所以改名字不會影響原本，但改權限會一起變。
```

C. Deep Copy: Create user_c by making a deep copy of base_template (using copy.deepcopy()). Change the name in user_c to "Charlie" and add "ADMIN" to its permissions list. Print both user_c and base_template, and explain why base_template remained completely unaffected.

```python
# (C)
base_template = ["User", ["READ"]]
user_c = copy.deepcopy(base_template)
user_c[0] = "Charlie"
user_c[1].append("ADMIN")

print("C - user_c:", user_c)
print("C - base_template:", base_template)

# 說明：
# 深層複製會複製所有層的內容。
# user_c 完全獨立，改它不會影響 base_template。
```

4. **(15%)** Given an integer n. Write a code snippet to create a list named primes that contains all prime numbers less than or equal to n.
   Requirements:
   A. Use a main for loop to iterate from 2 to n.
   B. Inside the main loop, you must use a nested for...else loop structure to determine if the current number is prime.

   Example:

```
# Input
n = 20

# After your code executes

# Expected content of the primes list
# [2, 3, 5, 7, 11, 13, 17, 19]
```

```
n = int(input())
primes = []
for num in range(2, n + 1):
    for i in range(2, num):
        if num % i == 0:
            break
    else:
        primes.append(num)
print(primes)
```

5. **(10%)** Given two sorted lists, list_a and list_b. Write a code snippet to merge them into a new, single sorted list called merged_list.
   Requirements:
   A. You must use a while loop to achieve this.
   B. Do not use the + operator to concatenate the lists and then sort them with sort().

   Example:

```
# Input
list_a = [1, 3, 5, 7]
list_b = [2, 4, 6, 8]

# After your code executes

# Expected content of merged_list
# [1, 2, 3, 4, 5, 6, 7, 8]
```

```
a_str = input()
list_a = []
for x in a_str.split():
    list_a.append(int(x))
b_str = input()
list_b = []
for x in b_str.split():
    list_b.append(int(x))
merged_list = []
i = 0
j = 0
while i < len(list_a) and j < len(list_b):
    if list_a[i] < list_b[j]:
        merged_list.append(list_a[i])
        i += 1
    else:
        merged_list.append(list_b[j])
        j += 1
while i < len(list_a):
    merged_list.append(list_a[i])
    i += 1
while j < len(list_b):
    merged_list.append(list_b[j])
    j += 1
for x in merged_list:
    print(x, end=" ")
print()
```

6. **(15%)** The "Look-and-Say" sequence is a sequence of digit strings generated by "reading out" the previous term.

   Generation Rule:

   A. The 1st term is "1".

   B. Starting from the 2nd term, each term is a description of the previous one.

   Example Breakdown:

   Term 1: "1"

   Term 2: You look at "1" and say "one 1", which gives "11".

   Term 3: You look at "11" and say "two 1s", which gives "21".

   Term 4: You look at "21" and say "one 2, one 1", which gives "1211".

   Term 5: You look at "1211" and say "one 1, one 2, two 1s", which gives "111221".

   Requirements:

   A. You must use a main loop to iterate n-1 times, generating each term from the first one.

   B. Inside the loop, you will need to scan the previous term's string to count consecutive identical digits to build the new term.

   Example:

```
# Input
n = 5

# After your code executes

# Expected string output
# "111221"
```

```
n = int(input())
term = "1"
for _ in range(1, n):
    new_term = ""
    i = 0
    while i < len(term):
        count = 1
        while i + 1 < len(term) and term[i] == term[i + 1]:
            count += 1
            i += 1
        new_term += str(count) + term[i]
        i += 1
    term = new_term
print(term)
```

7. **(10%)** Given a deeply nested list deep_list. Write a code snippet to "flatten" it into a single-dimensional list of numbers called flat_list.

   Requirements:

   A. Do not use recursion (a function calling itself).

   B. You must use a while loop and a temporary list to simulate stack behavior to process the elements.

   Example:

```
# Input
deep_list = [1, [2, 3], [4, [5, 6, [7]], 8], 9]

# After your code executes

# Expected content of flat_list
# [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
deep_list = eval(input())
flat_list = []
stack = deep_list[:]
while stack:
    element = stack.pop(0)
    if isinstance(element, list):
        stack = element + stack
    else:
        flat_list.append(element)
print(flat_list)
```