General Instructions

1. This activity consists of multiple short problems. Create one Python script file per problem.
2. Save each Python script according to the filename indicated in the problem statement.
3. **Each** Python script file should contain header information in comments, indicating your full name, your ID number, and the date you created your program.
4. **Each** Python script file should also contain **a comment block for the certification of authorship**, directly following the header information. See code_certification_template.py.
5. Create a folder named according to the convention: **HOA4-Surname-GivenName-IDNumber**. Place all of your Python files for this lab activity in this folder.
6. Archive your folder. On Windows, right-click on the folder and choose **Send To** > **Compressed (zipped) folder**. On a Mac, right-click on the folder and choose **Compress "folder name"**. This should produce a zip file named **HOA4-Surname-GivenName-IDNumber.zip**. Make sure that you created and named the folder correctly first **BEFORE** archiving.
7. Submit the zip file through the appropriate Moodle submission module.

Note: For this lab, there is **no need** to include a separate Certificate of Authorship document.

IMPORTANT:

- Do not produce any excess output (e.g. *The answer is...*)
- Do not print cues for input (e.g. *Please input a number:*)
- The format of your output must match the output specifications exactly (see **Sample Output** column for examples for each problem)
- Unless explicitly stated otherwise, assume that the user will always follow the input restrictions (e.g. if input n is described as `0 < n < 100`, then the user will always input a value within that range), so there is no need for you to check for those.

Problem A : Back Around
Filename : hoa4a.py
Description : Given a single word, take the last character of that word and print a new word with the last character added to the front and the back of the given word. The given word is guaranteed to have at least one character.
Input : Each test case is a line that contains one string, which is the word to process. Input is terminated by the word **STOP**.
Output : For each test case, print the new word on its own line. **For this problem, you have the option to produce the output immediately after the corresponding input.**

Sample Input	**Sample Output**

`cat`
`Hello`
`a`
`STOP`
	`tcatt`
	`oHelloo`
	`aaa`

Problem B : Even First
Filename : hoa4b.py
Description : Mr. Lee has a strange sequence of calling students for recitation in his class. He calls on the students with an even class number before he calls the ones with odd class numbers. Since he teaches computer science, class numbers start at `0`.
Input : Input consists of only one test case, a string corresponding to the names of the students, with one space separating each name. Names consist of only one word.
Output : Output consists of the sequence in which the students will be called for recitation. Print one name on each line.

Sample Input
```
Dorothy Tom Angela Ted Cheryl Sara Marsha
```

Sample Output
```
Dorothy
Angela
Cheryl
Marsha
Tom
Ted
Sara
```

Problem C : Print It to Win It
Filename : hoa4c.py
Description : A program that prints results of queries out of a collection of data
Input : The first line of input contains a line of words, each separated by a single space. These are the entries that comprise the data set.
The next line of input will contain queries, each also separated by a single space. The queries are integers.
Output : For each query i, print out the i-th word in the list: a query of `1` means that you must print the first word. A query of `5` means that you must print the fifth word, and so on. If a query is impossible, print **`NO`** to represent the frustration of the program.

Sample Input #1
```
apple banana cherry
2 1 8 2 3
```

Sample Output #1
```
banana
apple
NO
banana
cherry
```

Sample Input #2
```
car boat airplane
0 2
```

Sample Output #2
```
NO
boat
```

Problem D : Reversing String Order
Filename : hoa4d.py
Description : A program that prints a set of strings in reverse
Input : Input consists of an arbitrary number of test cases. Each test case starts with a line containing a number N ($1 \leq N \leq 20$).
The following N lines contain one string each. Input is terminated by a value of `-1`.
Output : For each test case, output the word **Case**, the case number (starting from **1**), and a **colon** in a single line. Then print N lines containing the given strings in reverse. **For this problem, you have the option to produce the output immediately after each test case.**

Note: You are not allowed to use any of the **reverse** commands. You are not allowed to use **negative slicing** (i.e. `[::-1]`).

Sample Input
```
2
Hello, World!
How are you?
1
This is just one line.
5
do
re
mi
fa
sus
-1
```

Sample Output
```
Case 1:
How are you?
Hello, World!
Case 2:
This is just one line.
Case 3:
sus
fa
mi
re
do
```

Problem E : Drawn Onward
Filename : hoa4e.py
Description : A program that determines whether a sequence is a palindrome
Input : Input consists of an arbitrary number of test cases.
Each test case is a single line that contains x number of integers ($1 \leq x \leq 1000$) separated by spaces. Each integer is positive, no greater than `99`.
Input is terminated by a single integer `0`.
Output : For each test case, output a single line. If the set of numbers in the test case is the same when read backwards (i.e., it is a palindrome), output **`Yes`**. Otherwise, output **`No`**. **For this problem, you have the option to produce the output immediately after each test case.**

Note: You are not allowed to use any of the **reverse** commands. You are not allowed to use **negative slicing** (i.e. `[::-1]`).

Hint: Palindromes are words or phrases (or sequences) that are the same when read backwards, ignoring spaces and punctuation marks, just like the title of this problem.

Sample Input #1
```
1 2 2 1
1 2 3 4 5
7
0
```

Sample Output #1
```
Yes
No
Yes
```

Sample Input #2
```
90 50 55 43 55 50 90
72 90 8 46 45
19 87 43 86 19
0
```

Sample Output #2
```
Yes
No
No
```

Reminders:

- Follow file naming conventions.
- Follow submission procedures.
- After submitting, double-check to see if you have successfully submitted the correct file.