# Data Structure and Programming Spring 2022
# Programming Assignment 1

Chi-Ting, Liu
b07901090@ntu.edu.tw

RELEASE DATE: 09:00 AM on 03/10/2022
DUE DATE: 04:00 AM on 03/24/2022

## 1 Problem Statement

This programming assignment asks you to read in a series of commands e.x. (PUSH 10, PUSH 9, POP, ......), and execute it using stack. We have written a code template for you, and *you should only fill in the* ***TODO*** *in stack.py and stack.cpp.*

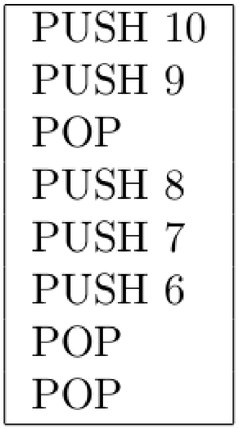
Notice:

- For stack.py, please use python3.7 to run, while the cpp version for stack.cpp is not limited.
- You should not modify the codes which are not specified by **TODO**.
- You should call the class **node()** when storing your data structure. That is, your data structure is limited to linked lists.
- You should be able to implement stacks with **O(1)**.
- For stack.cpp, you should avoid memory leak.

## 2 Input/Output Specification

### 2.1 Input Format

Inputs are a series of commands separated by newlines. The following is an example:

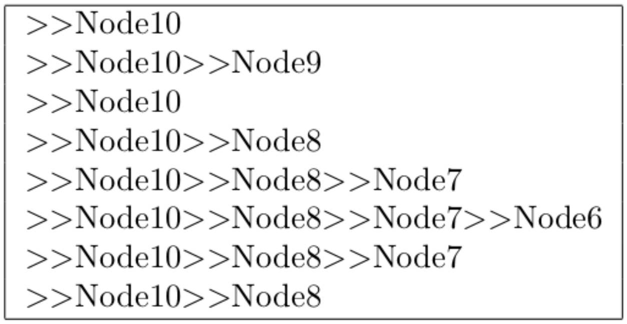


```
PUSH 10
PUSH 9
POP
PUSH 8
PUSH 7
PUSH 6
POP
POP
```

Figure 1: Input format

### 2.2 Output Format

You should print out your stack of each execution process. Below is an example of executing the above input using stack.



```
>>Node10
>>Node10>>Node9
>>Node10
>>Node10>>Node8
>>Node10>>Node8>>Node7
>>Node10>>Node8>>Node7>>Node6
>>Node10>>Node8>>Node7
>>Node10>>Node8
```

Figure 2: Output format

## 3 Submission

Please put stack.py and stack.cpp into a directory named studentID and compress the directory into studentID.zip. Finally, upload studentID.zip to NTU COOL.The homework is due on 3/24, at 04:00.

# 4 Evaluation

All of our test cases will not execute **pop()** on an empty stack. You won't need to handle this exception.

**Scoring Criteria:**

1. **Correctness and Time Complexity, 80%** : We will evaluate your code on five test cases. We provide you three of these test cases: input_1.txt, input_2.txt, and input_3.txt, and their corresponding golden output: golden_1.txt, golden_2.txt, and golden_3.txt. We also provide a script: evaluation.sh so that you can check whether your programs pass these test cases. Furthermore, this script will display the runtime. If your implementation is correct, you will find that compiled executable program runs much faster. If one of your programs can pass one test case with time complexity O(1), you will get 8% of the total score. Since you have two programs: stack.py and stack.cpp, running on 5 test cases, if both of your programs pass all of them with time complexity O(1), you will get 80% of the total score.

2. **Memory Management, 20%** : After you got the above 40% score of your cpp program, we will evaluate it on two test cases for memory management. We provide you one of these testcases: input_mem_1.txt. You can use "System Monitor" to check whether your cpp program has memory leak. If your program has no memory leak, you can get 10% for each test case, and get 20% for all test cases.