# Data Structure and Algorithm, Spring 2022

# Homework 1

**Due: 13:00:00, Tuesday, March 29, 2022**

TA E-mail: dsa_ta@csie.ntu.edu.tw

**March 8 Update: The time range 10 quota is given.**

## Rules and Instructions

- Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.
- In Homework 1, the problem set contains a special Problem 0 (see below) and 5 other problems. The set is divided into two parts, the non-programming part (Problems 0, 1, 2, 3) and the programming part (Problems 4, 5).
- For problems in the non-programming part, you should combine your solutions in *one* PDF file. Your file should generally be legible with a white/light background—using white/light texts on a dark/black background is prohibited. Your solution must be as simple as possible. At the TAs' discretion, solutions which are too complicated can be penalized or even regarded as incorrect. If you would like to use any theorem which is not mentioned in the classes, please include its proof in your solution.
- The PDF file for the non-programming part should be submitted to Gradescope as instructed, and you should use Gradescope to tag the pages that correspond to each subproblem to facilitate the TAs' grading. Failure to tagging the correct pages of the subproblem can cost you a 20% penalty.
- For the programming part, you should have visited the *DSA Judge* (https://dsa-2022.csie.org) and familiarized yourself with how to submit your code via the judge system in Homework 0.
- For problems in the programming part, you should write your code in C programming language, and then submit the code via the judge system. In each day, you can submit up to 5 times for each problem. To encourage you to start early, we allow 10 times of submissions per day in the first week **(2022/03/08-2022/03/15)**. The judge system will compile your code with

```
gcc main.c -static -O2 -std=c11
```

- Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.
- Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.
- The score of the part that is submitted after the deadline will get some penalties according to the following rule:

$$\text{LateScore} = \max\left(0, \frac{86400 \cdot 5 - \text{DelayTime (sec.)}}{86400 \cdot 5}\right) \times \text{OriginalScore}$$

- If you have questions about HW1, please go to the discord channel and discuss (*strongly preferred*, which will provide everyone a more interactive learning experience). If you really need an email answer, please follow the rules outlined below to get a fast response:
  - The subject should contain two tags, `"[hw1]"` and `"[Px]"`, specifying the problem where you have questions. For example, `"[hw1][P5] Is k in subproblem 5 an integer"`. Adding these tags allows the TAs to track the status of each email and to provide faster responses to you.
  - If you want to provide your code segments to us as part of your question, please upload it to Gist or similar platforms and provide the link. Please remember to protect your uploaded materials with proper permission settings. Screenshots or code segments directly included in the email are discouraged and may not be reviewed.

## Non-Programming Part

# Problem 0 - Proper References (0 pts)

For each problem below, please specify the references (the Internet URL you consulted with or the classmates/friends you discussed with) in your PDF submitted to Gradescope. *You do not need to list the TAs/instructors.* If you finished any problem all by yourself (or just with the help of TAs/instructors), just say "all by myself." While we did not allocate any points on this problem, failure to complete this problem can lead to penalty (i.e. negative points). Examples are

- Problem 1: Alice (B86506002), Bob (B86506054), and https://stackoverflow.com/questions/982388/
- Problem 2: all by myself
- ...

Listing the references in this problem does *not* mean you could copy from them. We cannot stress this enough: *you should always write the final solutions alone and understand them fully.*

# Problem 1 - Analysis Tools (100 pts)

Suppose that $f$ and $g$ are asymptotically non-negative functions defined on $\mathbb{N}$. Recall the following asymptotic notations:

- $f(n) = O(g(n))$ if and only if there exist positive numbers $c$ and $n_0$ such that

$$f(n) \leq cg(n) \text{ for all } n \geq n_0.$$

- $f(n) = \Omega(g(n))$ if and only if there exist positive numbers $c$ and $n_0$ such that

$$f(n) \geq cg(n) \text{ for all } n \geq n_0.$$

- $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Read the following pseudo code carefully. Write down the time complexity for each of the following functions using the $\Theta$ notation along with an expression of $n$. Two requirements:

- Your answer should be as simple as possible. For instance, $\Theta(\frac{1}{n} + \lg(n^{543}) + 8763n^2)$ should be written as $\Theta(n^2)$.
- Your answer should also come with a *brief* explanation. The word *brief* means "very short." An example of a *brief* explanation is as follows.

  Sample problem: Write down the time complexity for this function using the $\Theta$ notation along with an expression of $n$.

  HELLO-WORLD($n$)
  1 **for** $i = 1 \,..\, n$
  2 **for** $j = 1 \,..\, 2n$
  3 print("Hello world")

  Sample solution: The iteration counts of the two loops are $n$ and $2n$ respectively, so there are $2n^2$ iterations. Hence the time complexity is $\Theta(n^2)$. □

1. (15 pts)

   FUNC-A($n$)
   1 $sum = 0, i = 1$
   2 **while** $sum < n$
   3 $sum = sum + i$
   4 $i = i \times 2$

2. (15 pts)

```
FUNC-B(n)
1  if n > 0
2      FUNC-B(n − 1)
3      FUNC-B(n − 1)
4      FUNC-B(0)
```

3. (20 pts)

```
FUNC-C(n)
1  i = n
2  while i > 1
3      j = 0
4      while j < i
5          j = j + 1
6      i = i/2 // We will simply consider division with real numbers
```

Next, prove or disprove each of the statement below. You should provide a *formal proof*. That is, you need to specify all constants like $c$ and $n_0$ or use any theorems that have been proved in class if you believe the statement to be true, or provide a counterexample if you believe the statement to be false.

- Sample problem: Prove that $an + b = \Theta(n)$, where $a$ and $b$ are positive constants.
- Sample full-credit formal proof 1: Observe that $an \leq an + b \leq 2an$ for $n \geq \lceil \frac{2b}{a} \rceil$, so $an + b = \Theta(n)$. □
- Sample full-credit formal proof 2: Rewrite $an + b = n(a + bn^{-1})$. Obviously we have $\lim_{n\to\infty} \frac{an+b}{n} = a > 0$. Using the proof in class that shows the limit definition leads to the formal definition, we establish that $an + b = \Theta(n)$. □

4. (15 pts) (Prove or disprove) If $f(n) = O(g(n))$, then $f(n) \cdot g(n) = \Omega((f(n))^2)$

5. (15 pts) (Prove or disprove) For any $k > 0$, $\lg n = O(n^k)$. You can use the fact that $\ln n \leq n - 1$ for $n \geq 1$ if needed. Here lg means $\log_2$ and ln means $\log_e$.

6. (20 pts) (Prove or disprove) If $f(n) = O\big(g(n)\big)$, $\lg f(n)$ and $\lg g(n)$ are asymptotically non-negative functions, and there exists some $n_2$ such that $g(n) \geq 2$ for all $n \geq n_2$, then $\lg f(n) = O\big(\lg g(n)\big)$.

The following three subproblems are for your own practice. You do not need to submit them. The conclusions might be used for future classes, and some similar problems might show up in exams as well. You can basically use what you have learned in calculus classes to help you solve the subproblems.

7. (0 pts, self-practice) (Prove or disprove) $\sum_{k=1}^{n} \frac{n}{k} = \Theta(n \lg n)$.

8. (0 pts, self-practice) (Prove or disprove) $\lg(n!) = \Theta(n \lg n)$.

9. (0 pts, self-practice) (Just prove) For this subproblem, consider functions $f(x)$ and $g(x)$ defined on any non-negative real numbers instead of $\mathbb{N}$, such that $f(x) = O(g(x))$. Now, consider a new function $h(x)$ which satisfies the following properties:

   - The function $h(x)$ maps non-negative real numbers to non-negative real numbers.
   - The function $h(x)$ is monotonically increasing: $h(x') \geq h(x)$ for all $x' \geq x$.
   - The inverse function $h^{-1}(y)$ of $h$ exists for any $y > 0$.

   Then, prove that $f(h(x)) = O(g(h(x)))$.

   You can also challenge yourself by removing some or all the properties above and see if you can construct any interesting counterexamples.

# Problem 2 - Stack / Queue (100 pts)

## Ground Work

In this subsection, the answers are directly related to what we have taught in class.

1. (15 pts) Convert the infix expression below into a postfix expression, with the usual precedence of () over $\times/$ over $+-$ and left association. Briefly explain the *human algorithm* (which is effective for human brains but does not necessarily need to be effective for computers) that you use for conversion.

$$((1+2)\times(3+4\times5))+(9/3+7\times5)$$

2. (15 pts) Convert the postfix expression below into an infix expression with the same precedence as the previous subproblem. Briefly explain the *human algorithm* that you use for conversion.

$$1, 2, +, 5, 3, -, \times, 6, \times, 5, /$$

   Then, evaluate the outcome. Please treat / as the usual division in math, and express the outcome as an irreducible fraction.

3. (15 pts) Consider an empty queue implemented by a circular array $a$ of size 4 whose indices start at 0. During the execution of a sequence of enqueue/dequeue operations, please fill out the internal state of $a$ in the table below. No other explanations are needed for this subproblem.

| | $a[0]$ | $a[1]$ | $a[2]$ | $a[3]$ |
|---|---|---|---|---|
| enqueue 1 (example) | 1 | NIL | NIL | NIL |
| enqueue 5 | | | | |
| enqueue 3 | | | | |
| dequeue | | | | |
| enqueue 4 | | | | |
| enqueue 6 | | | | |
| dequeue | | | | |

4. (15 pts) Genius-mahjong-saki is a competitive mahjong-player in NTU CSIE. She always plays mahjong—conducts experiments to study probability—during the DSA class time ( seriously?! :-) ). Since she is a genius, she never arranges her mahjong hands. In each round, she puts the new tile in the rightmost of her hand and discards the leftmost tile. Assume that when the play begins, her hand (13 tiles) from left to right is

$$4p, 2p, 9p, 8p, 7p, 6p, 5p, 4p, 3p, 2p, 1p, 1p, 1p$$

If she wants her hand from left to right to become

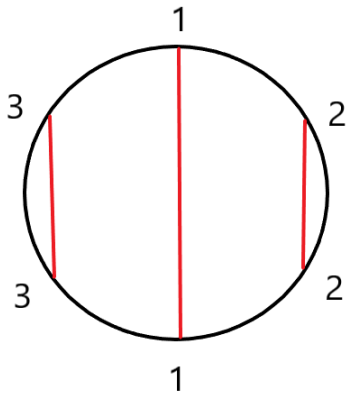
$$1p, 1p, 1p, 2p, 3p, 4p, 5p, 6p, 7p, 8p, 9p, 9p, 9p$$

to win chuuren poutou (九蓮寶燈), what is the smallest number of rounds needed? Briefly explain your answer.

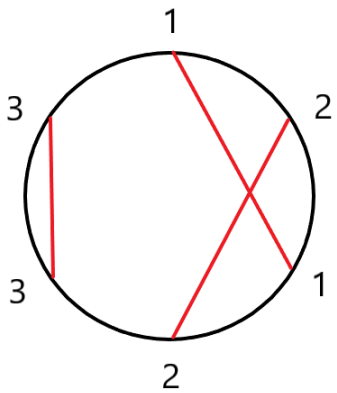
## Booster

In this subsection, the subproblems require some deeper thinking. You are encouraged to not just think alone, but also discuss with other classmates (legally) and/or come to the TA hours. Good luck!

Assume that $n$ couples (labeled from 1 to $n$) are standing around a circle. As valentine's day is coming, they want to go on a date with their sweetheart. Joy, the god of marriage (月老) in CSIE, is tying red cords (牽紅線) between the couples. The red cords might entangle with each other if they intersect. Thus, Joy needs to carefully check if there are any such intersections. Given $n$ couples' position clockwise, please help Joy to check if the red cords intersect!



[1,2,2,1,3,3]



[1,2,1,2,3,3]

For example, as shown above, if the couples, labeled $[1, 2, 2, 1, 3, 3]$ respectively, are arranged clockwise on the circle, then Joy can draw segments (on a circle) between the two labeled 1, the

two labeled 2, and the rest two labeled 3, such that the red cords do not intersect. However, if the couples are labeled $[1, 2, 1, 2, 3, 3]$ respectively, the red cords 1 and 2 would intersect.

5. (25 pts) Given the labels of the $n$ couples on the circle, please design an $O(n)$-time algorithm to check if the red cords intersect. You may assume that the labels are initially stored in a length-$n$ array. Briefly explain the correctness of your algorithm. (*Hint: Use stacks*)

6. (15 pts) Prove that your algorithm meets the time complexity requirement.

# Problem 3 - Array / Linked Lists (100 pts)

For all subproblems below, your answers should include the following parts:

(a) Describe your algorithm with pseudo code and briefly explain why it works.

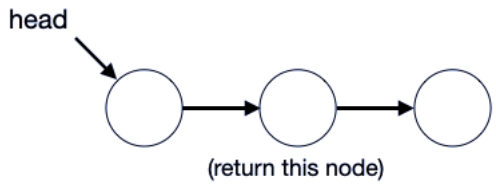
(b) Analyze the time complexity and *extra-space* complexity of your algorithm. If your algorithm is not efficient enough in time or in space, you may not get full points.

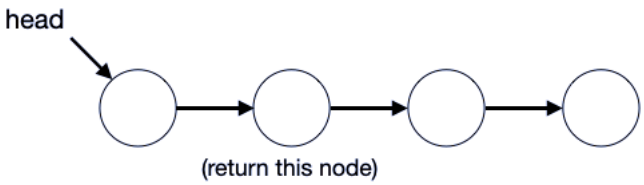
It is rumored that there is a forbidden library lying under the Main Library (總圖). People who read the secret books stored inside can become the master of algorithm. To reach the destination, you are asked to solve several puzzles first.

1. (30 pts) For the first puzzle, the only way to go underground in the Main Library is to first go to the top of the bell tower to access a transmission portkey. Somehow there is a barrier in front of the tower, and you can only bypass the barrier by climbing up the tower with a magic chain. The magic chain that you have on hand is a bit fragile. Thus, you decide to strengthen the chain by folding. That is, you need to find the middle point of the magic chain.

   The DSA wizard gave you the magic chain as a linked list $L$ with $L.head$ as its head. Furthermore, the wizard cast an evil spell on the magic chain. Whenever one sees the link to the tail node of the chain, $L.head$ disguises itself and becomes invalid to access. That is, you cannot revisit the list again after walking through the chain from the head to the tail. The spell eliminates the possibility of a simple algorithm that counts the number of nodes in the chain in the first pass and then finds the middle point in the second pass.

   Under such an evil spell, please design an algorithm that finds the middle node with $O(n)$-time and $O(1)$-extra-space, where $n$ is the length of the linked list that you *do not* know on hand.

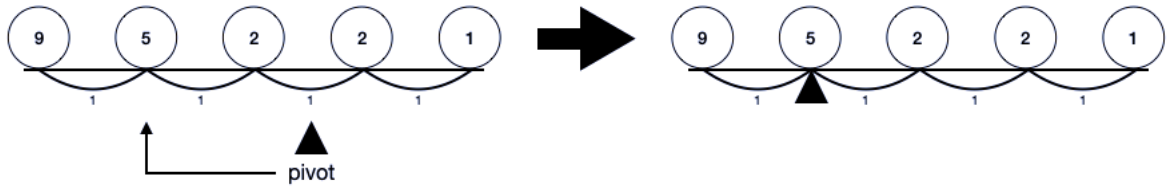


2. (30 pts) For the second puzzle, there is a huge gate standing in front of you, with $n$ *positive integers* written on it. The troll at the gate asks you to find the first missing number. For example, if the numbers are $[5, 2, 1, 1, 2, 3, 5, 8]$, then the answer would be 4. Note that the numbers, as shown above, are *not* guaranteed to be sorted.

   Given $n$ such numbers stored in an unsigned integer array, please design an algorithm that finds the first missing number with $O(n)$-time and $O(1)$-extra-space.

3. (40 pts) Here comes the final puzzle. There is a see-saw with $n$ stones on it. Each stone has an integer weight denoted by $w_i$ $(i = 1, 2, \ldots, n)$, and the distance between any two adjacent stones is 1 unit. Now you are asked to put the pivot under one of the stones to balance the see-saw. For example, if we put the pivot under stone 2 below, then the total torque (力矩) would be: $(9 \times 1) - (2 \times 1 + 2 \times 2 + 1 \times 3) = 0$. That is, the see-saw would be balanced.




Given $n$ such weights stored in an integer array, please design an algorithm that finds the smallest location to place the see-saw-balancing pivot with $O(n)$-time and $O(1)$-extra-space.

# Problem 4 - Calculate Simple Integers Easily [CSIE] (100 pts)

## Problem Description

Poor calculation bothers Lisa's life. One day, she learned how to implement an easy calculator in the DSA course. She decided to implement it to conquer her arithmetic obstacles immediately. The calculator supports the following operator on long long integers with correct precedence (*Hint:* *https://en.cppreference.com/w/cpp/language/operator_precedence*)

- arithmetic operators: `+`, `-`, `*`, `/`, %, where / means integer division. That is, $a/b = \lfloor \frac{a}{b} \rfloor$.
- parentheses: `(`, `)`

It is guaranteed that

- the divisor for / and % would not be zero during the process of the calculation;
- $a > 0$, $b > 0$, for all operations $a\%b$.

To be more convenient, Lisa's calculator should support continual arithmetic. More specifically, Lisa's calculator follows the rules:

- At the beginning, the register of Lisa's calculator is empty.
- When Lisa's calculator reads any character except '=', it appends the character to the end of the register.
- When Lisa's calculator reads the '=' character, it calculates the answer given by the expression in the register with the correct precedence. It then prints out the answer and replaces the register's entire content with the answer.

For example, assume that Lisa inputs

```
1+2=*5+(1+7+11)=%5-11=
```

The desired outputs are

| output | reason |
|---|---|
| `3` | $(1 + 2 = 3)$ |
| `34` | $(3 * 5 + (1 + 7 + 11) = 34)$ |
| `-7` | $(34\ \%\ 5 - 11 = -7)$ |

This kind of functionality is common in many utility programs. You can use those programs to help you generate test data and debug by yourself. But you should of course solve this problem *within your C code without calling any external programs* in order to pass our judge system!

## Input

The input contains a line, recording Lisa's input.

## Output

For each '`=`' in the input line, output a line of '`Print: [answer]`' with the answer being the calculated answer from the expression in the register.

## Constraints

- $0 <$ the length of the input line $< 10^6$
- $0 < a_i < 10^8$ for each number $a_i$, which is guaranteed to be an integer, in the input.
- Every mathematical operation will not exceed the range of long long int.
- When encountering '`=`', the expression in the register is guaranteed to be legal.

### Subtask 1 (25 pts)

- the operators include only arithmetic operators, and only one `=`

### Subtask 2 (25 pts)

- all operators, but only one `=`

### Subtask 3 (50 pts)

- all operators and multiple `=`

## Sample Cases

### Sample Input 1

```
1+3-2*4=
```

### Sample Output 1

```
Print: -4
```

### Sample Input 2

```
1+2*(3+(8-6/3)*5)=
```

### Sample Output 2

```
Print: 67
```

### Sample Input 3

```
1+(2*3*(1+2+3*4/2*7+(1+8))+4)=/(11/5)+7-19*2=
```

### Sample Output 3

```
Print: 329
Print: 133
```

# Problem 5 - Purple Cow (100 pts)

## Problem Description

There is a mystery in Alley 118.

`Never order a medium rare steak in Purple Cow (紫牛).`

After celebrating the birthday in *Purple Cow*, USA has a stomachache. He starts to look for a bathroom. However, it is the peak time for bathrooms, and thus there are lots of people waiting in multiple lines. After waiting in a line for a while, USA notices that the number of the people in front of him increases. Afterwards, he discover that they are all impolite USBs who would cut in line and stand after their friends if they have friends from the *same group* that are already in the line.

There are $M$ bathrooms with indices $0, \ldots, M-1$, and $K$ groups of students with indices $0, \ldots, K-1$. Now you, a DSA STUDENT (Super Toilet & Unbelievably Diligent & Extra Noted Technician), would need to help USA simulate how these lines change. The following 4 situations might happen and cause some changes to the lines.

- `enter` $i$ $j$ $m$

  USB with id $j$ from group $i$ comes to stand in line $m$ (the line for bathroom $m$). Note that the USB is going to cut into the position after the *last* person in group $i$. If no one in group $i$ is in that line, the USB is going to stand at the end of the line.

- `leave` $m$

  The last USB in line $m$ walks away.

- `go` $m$

  The first USB in line $m$ goes to use bathroom $m$.

- `close` $m$

  Bathroom $m$ is out of toilet paper and closes. So USBs in that line would move to the *nearest* open bathroom with a smaller index. Since the bathroom is placed circularly for some unknown reasons (probably because the earth is round :-) ), USBs would search for the first open bathroom in $M-1, M-2, \ldots$ if all the smaller-indexed bathrooms have been closed.

  When moving, USBs from the end of line $m$ would move first because they are closer to the end of the other line. So effectively the order in line $m$ is reversed in the new line. Besides, all those USBs still keep the habit of cutting in like `enter`ing when moving to another line.

## Input Format

The first line contains three integers $M$, $N$, $K$, representing $M$ bathrooms, $N$ situations and $K$ groups. The next $N$ lines are formatted as follows.

- `enter`, followed by three integers `i j m` separated by spaces
- `leave`, followed by one integer `m`
- `go`, followed by one integer `m`
- `close`, followed by one integer `m`

## Output Format

You should output $M$ lines, with line $m$ containing the ids of the USBs waiting for bathroom $m$ by their order in the line.

## Constraints

1. $1 \leq M, N, K \leq 10^6$
2. $1 \leq M \cdot K \leq 10^6$
3. every USB's id is *distinct* and ranges in $[1, 10^8]$
4. There will be no invalid situations, such as entering/closing a closed bathroom or leaving/going from an empty line.

## Subtasks

### Subtask 1 (20 pts)

- $M = 1$
- $N, K \leq 10^3$
- no `close` situation

### Subtask 2 (20 pts)

- $1 \leq M, N, K \leq 10^3$
- $1 \leq M \cdot K \leq 10^4$
- no `close` situation

**Subtask 3 (20 pts)**

- $1 \leq M, N, K \leq 10^3$
- $1 \leq M \cdot K \leq 10^4$

**Subtask 4 (40 pts)**

- no other constraints

## Sample Cases

**Sample Input 1**

```
1 11 4
enter 0 1 0
enter 1 2 0
enter 0 3 0
enter 1 4 0
enter 2 5 0
enter 1 6 0
leave 0
leave 0
go 0
go 0
enter 0 7 0
```

**Sample Output 1**

```
2 4 7
```

**Sample Input 2**

```
3 14 4
enter 0 1 0
enter 1 2 0
enter 0 3 0
enter 2 4 1
enter 1 5 1
enter 0 6 1
enter 1 7 1
enter 3 8 1
enter 1 9 2
enter 0 10 2
close 1
go 0
close 0
go 2
```

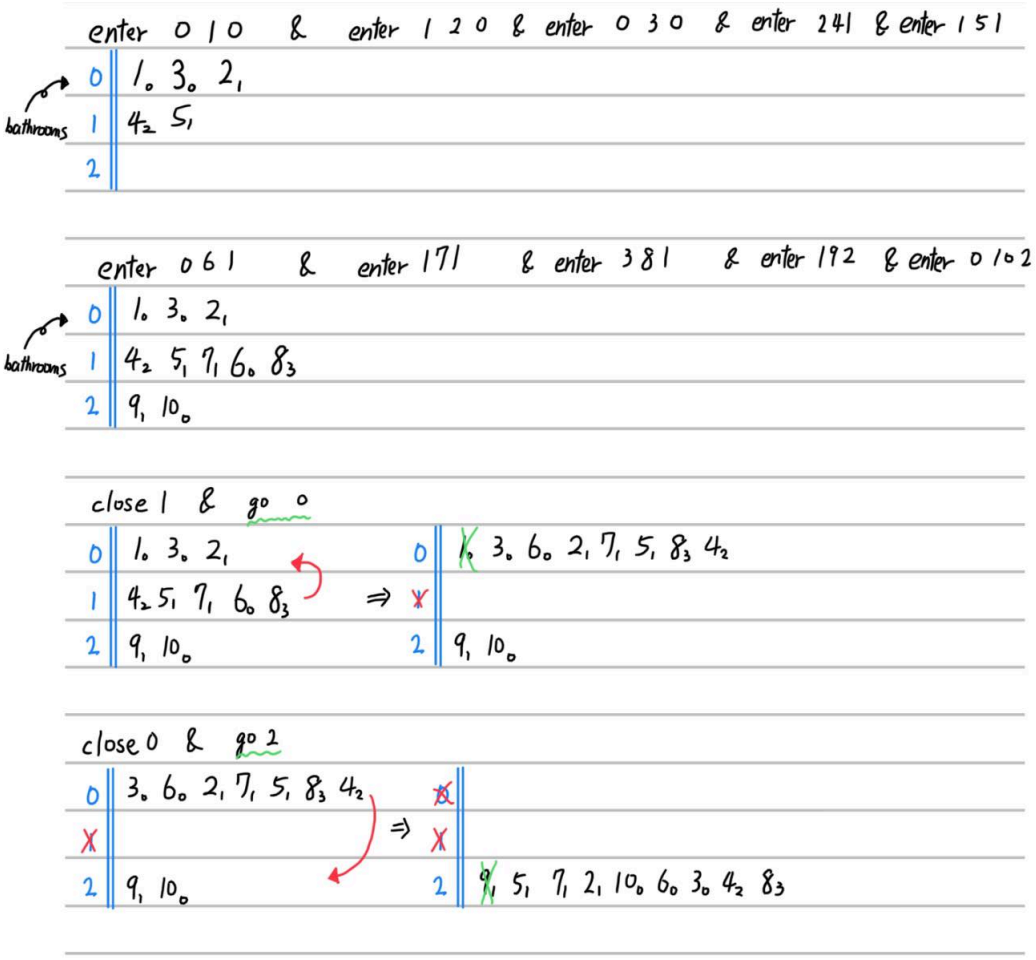
**Sample Output 2**

```


5 7 2 10 6 3 4 8
```

Note that there are two empty lines above `5 7 2 10 6 3 4 8`. For more specific explanations, please refer to the demo below.

**Sample 2 Demo**




## Hints & Notes

- Closing an empty bathroom is valid situation.
- Moving USBs one by one is possibly too time-consuming.
- All you need are the data structures taught in class. But if you are interested in using even more advanced data structures to solve this problem, consider searching with the following keywords: *reversing doubly-linked list in* $O(1)$.
- Go⊕d Luck & Have Fun!