Homework #3

Stacks and Queues

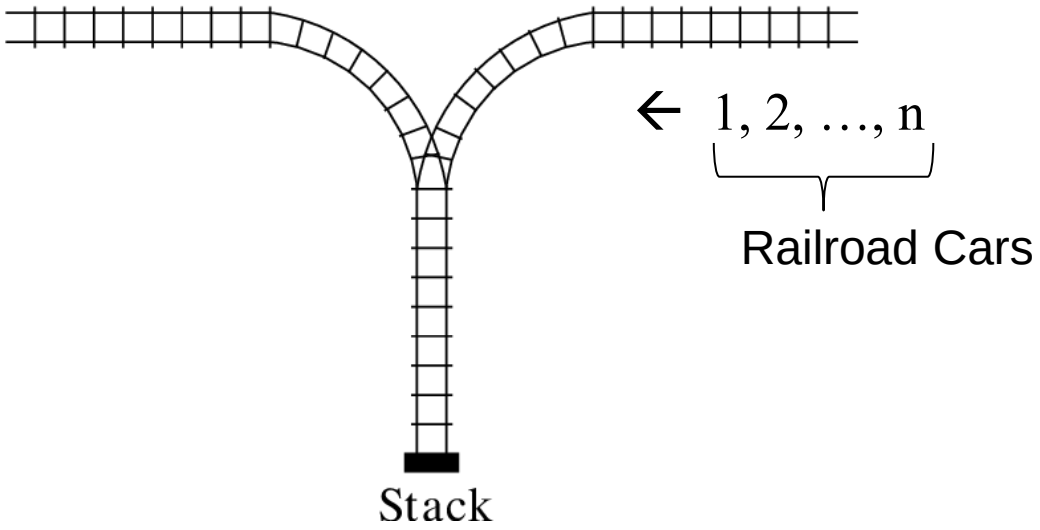
Yunmin Go

School of CSEE

HANDONG GLOBAL UNIVERSITY

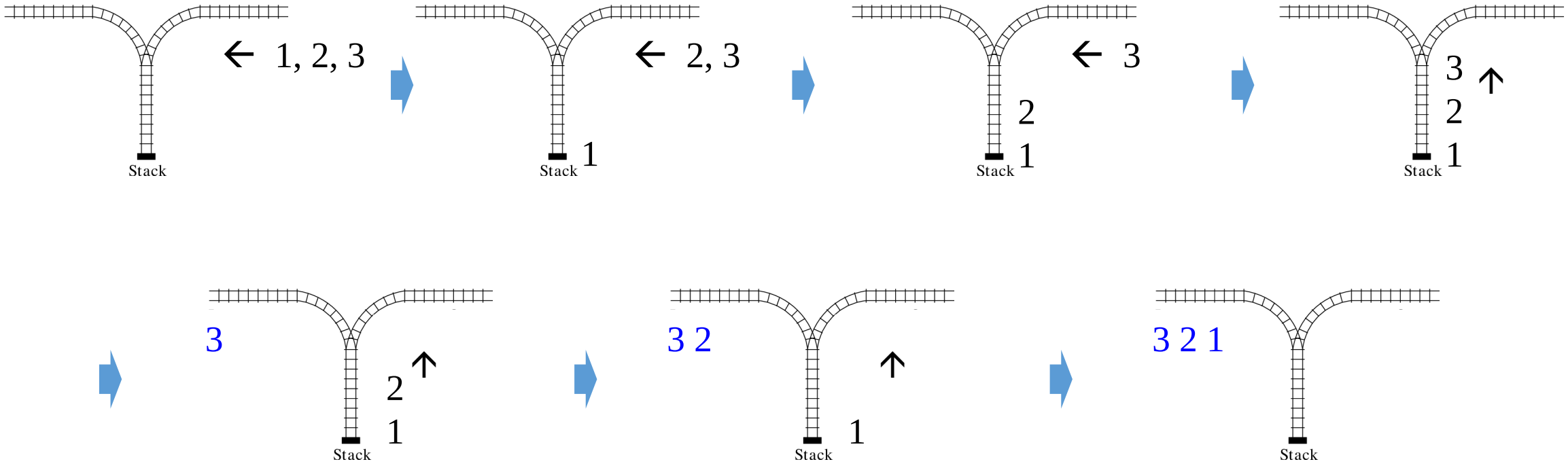
HW#3-1

- Consider the railroad switching network in figure below. Railroad cars numbered 1, 2, …, n are at the right. Each car is brought into the stack and removed at any time. For instance, if n=3, we could move in1, move in 2, move in 3, and then take the cars out, producing the new order 3, 2, 1. For other example, we could also move in 1, move in 2, take 2 out, move in 3, take 3 out, take 1 out, producing the new order 2, 3, 1. Write the C++ program that evaluates whether the order of cars entered by user is the possible permutations or not.



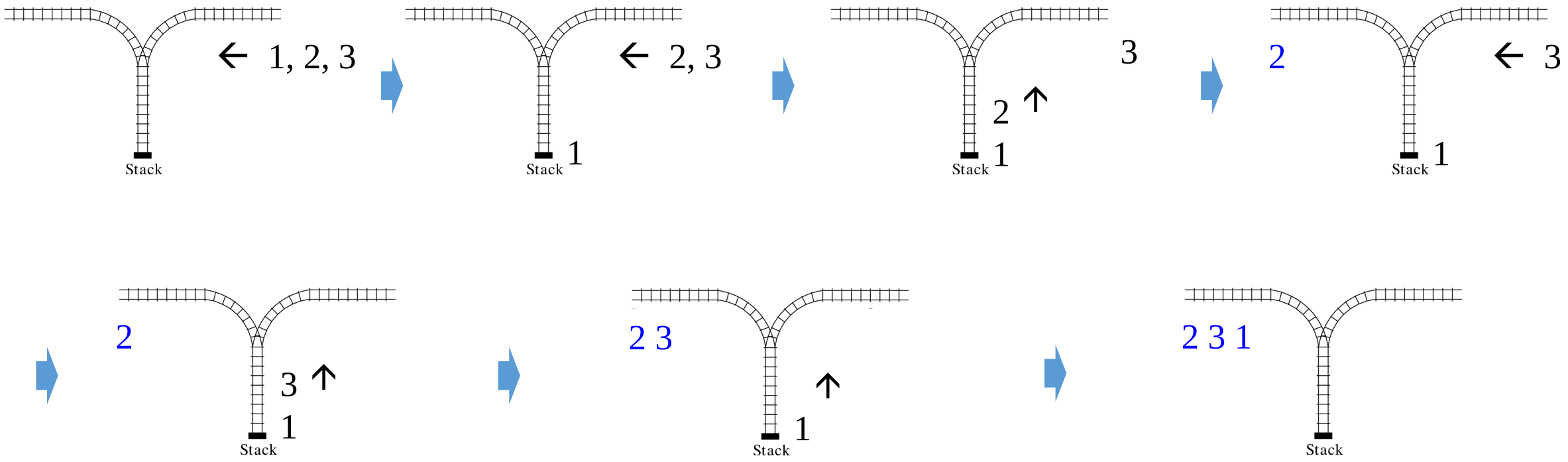
HW#3-1

- Example #1
  - We could move in 1, move in 2, move in 3, and then take the cars out, producing the new order 3, 2, 1.



HW#3-1

- Example #2
  - We could also move in 1, move in 2, take 2 out, move in 3, take 3 out, take 1 out, producing the new order 2, 3, 1.



HW#3-1

- Program procedures
  - User enters the number of cars.
  - User enters the expected order of cars.
  - If the entered order of cars is the possible permutations, the program prints out “Possible”. Otherwise, the program prints out “Not possible”.
  - Repeat above procedures until user enters 0 (zero).

- Use your stack and queue class implemented for practice session.
  - You can modify your stack and queue classes if you want.

- The main file name should be railroad.cpp.

HW#3-1

- Expected results

```
PS C:\ds\hw03> .\railroad.exe
Enter the number of cars: 3
Enter the expected order (1~3): 3 2 1
Input: 123
Output: 321
Possible
Enter the number of cars: 3
Enter the expected order (1~3): 2 3 1
Input: 123
Output: 231
Possible
Enter the number of cars: 3
Enter the expected order (1~3): 2 1 3
Input: 123
Output: 213
Possible
Enter the number of cars: 3
Enter the expected order (1~3): 3 1 2
Input: 123
Output: 312
Not Possible
```

```
Enter the number of cars: 4
Enter the expected order (1~4): 2 3 4 1
Input: 1234
Output: 2341
Possible
Enter the number of cars: 4
Enter the expected order (1~4): 2 4 3 1
Input: 1234
Output: 2431
Possible
Enter the number of cars: 4
Enter the expected order (1~4): 4 2 3 1
Input: 1234
Output: 4231
Not Possible
Enter the number of cars: 0
PS C:\ds\hw03>
```

HW#3-2

- Write a program that evaluates the infix expression having unary operators + and –.

- Examples

Infix: +2 * –3 + 6 – –7

Postfix: 2 –3 * 6 + –7 –

Evaluation: 7

Infix: –3 * 2 – –9 / –3 ++4 – –2

Postfix: –3 2 * –9 –3 / – 4 + –2 –

Evaluation: –3

HW#3-2

- Refer to p.46~67, Chapter 3.
- Assume that the operands are one digit signed integer numbers.
- Consider the operators used in *get_token* function in p.55
- Transform an infix expression into a postfix expression and evaluate the postfix expression.

HW#3-2

- The infix expression is passed by command line arguments without space when user runs the program.

- The main file name should be infixeval.cpp.

HW#3-2

- Expected results

```
PS C:\ds\hw03> .\infixeval.exe "+2*-3+6--7"
Infix: +2*-3+6--7
Postfix: 2 -3 * 6 + -7 -
Evaluation: 7
PS C:\ds\hw03> .\infixeval.exe "+2*(-3+6)--7"
Infix: +2*(-3+6)--7
Postfix: 2 -3 6 + * -7 -
Evaluation: 13
PS C:\ds\hw03> .\infixeval.exe "-3*2--9/-3++4--2"
Infix: -3*2--9/-3++4--2
Postfix: -3 2 * -9 -3 / - 4 + -2 -
Evaluation: -3
PS C:\ds\hw03> .\infixeval.exe "-3*(2--9)/-3++4--2"
Infix: -3*(2--9)/-3++4--2
Postfix: -3 2 -9 - * -3 / 4 + -2 -
Evaluation: 17
```

Common Requirements

- HW#3-1 is 60%. HW#3-2 is 40%
- You don’t need to use class, except for stacks and queues.
- All of C-style functions and headers are allowed
  - E.g., printf, fopen, fgets, etc.
- Write clean source code
  - Add proper comment in your source code
  - Consider code indentation for enhancing readability

Common Requirements

- Upload ZIP file on LMS by compressing all your source codes (including stacks and queue classes)
  - File name: hw03_student id.zip (ex: hw03_20400022.zip)

- Due date: 11pm, 4/20 (Tue)